



Escola Politècnica Superior  
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# TRABAJO FINAL DE CARRERA

**Título:** Diseño de una estación meteorológica autónoma gestionada mediante mensajes SMS

**Titulación:** Ingeniería Técnica de Telecomunicaciones, especialidad Sistemas de Telecomunicaciones

**Autor:** Jorge Luján de Francisco

**Director:** Oscar López Lapeña

**Fecha:** 4 de diciembre de 2008

**Título:** Diseño de una estación meteorológica autónoma gestionada mediante mensajes SMS

**Autor:** Jorge Luján de Francisco

**Director:** Oscar López Lapeña

**Fecha:** 4 de diciembre de 2008

## **Resumen**

El objetivo del TFC es el diseño de un sistema electrónico para la adquisición y transmisión de los parámetros físicos típicos de una estación meteorológica.

Los parámetros meteorológicos medidos han sido elegidos por su fácil interpretación por un usuario, a fin de detectar posibles lluvias, niebla, heladas, riesgo de incendios, etc., todo ello sin necesidad de desplazarse al lugar del emplazamiento.

La estación meteorológica puede ser emplazada en cualquier lugar, tanto en zonas urbanas como en rurales, o campos, donde no se dispone de red eléctrica para su alimentación ni una red de comunicaciones por cable.

Dicho proyecto consta de un desarrollo de hardware que engloba el diseño de los circuitos de captación, el acondicionamiento de los sensores, la comunicación con el sistema GSM, la unidad central de procesamiento y la alimentación.

El software desarrollado para la estación meteorológica controla la gestión de los usuarios, los niveles de batería, los sensores y las comunicaciones.

Como características principales cabe destacar:

- El sistema de alimentación autónomo mediante paneles solares y baterías. A fin de poder emplazar el sistema en cualquier ubicación y su autorregulación para disminuir el consumo.
- El sistema de comunicaciones inalámbrico, vía red de comunicaciones GSM instaladas por las compañías telefónicas. Consiguiendo una movilidad total para su emplazamiento.
- La facilidad para los usuarios a la hora de enviar sms a fin de conocer alguno de los parámetros de la estación meteorológica y establecer parámetros de alarma.

El objetivo final es la gestión eficaz de los recursos del sistema a fin de lograr un funcionamiento simple, autónomo y de un tamaño reducido para su libre ubicación.

**Title:** Design of a weather station autonomously managed through SMS messages

**Author:** Jorge Luján de Francisco

**Director:** Oscar López Lapeña

**Date:** December, 4 2008

## Overview

The target of the TFC is the design of an electronic device in order to establish the acquisition and administration of physical parameters typical of a weather station.

The weather parameters measured have been chosen due to their easy interpretation by the user, so that they can detect possible rains, fogs, snowstorms, risks of fire, etc. everything without the necessity of moving towards the place itself.

The weather station could be situated in any place, not only in urban but also in rural areas, where there are no supply systems.

This project consists of the development of the hardware (which involves the desing of circuits to pick up signals), the conditioning of sensors, the communication using the GSM system, the central unit of processor and supply.

The type of software developed for this weather station deals with the management by the users, the levels of battery, the sensors and communications.

As the main features, we will point out:

- The autonomous supply system using solar panels and batteries. In order to be able to situate the system in any place and its autocontrol to decrease the consumption.
- The wireless communication system, suing the GSM supply system installed by the telephone companies. This way we will achieve a total mobility to its location.
- The easiness for users when sending an sms so that they know some of the parameters of the weather station and to establish alarm parameters.

The final targets are, on the one hand, the correct administration of the system resources so as to achieve the simple and autonomous use, and on the other hand, the reduce size of the device so that makes its location free.

# ÍNDICE

<b>CAPÍTULO 1. INTRODUCCIÓN.....</b>	<b>6</b>
1.1. Objetivo .....	6
1.2. Planteamiento del proyecto.....	7
<b>CAPITULO 2. UNIDAD CENTRAL DE CONTROL .....</b>	<b>9</b>
2.1 Introducción a los microcontroladores.....	9
2.2 Elección del microcontrolador .....	10
2.2 Programación mediante MPLAB.....	13
<b>CAPITULO 3. DESARROLLO DEL PROGRAMA.....</b>	<b>15</b>
3.1 Funciones del microcontrolador en el sistema.....	15
3.2 Definición de las entradas y salidas del PIC .....	15
3.3 La programación de la estación meteorológica.....	16
3.3.1 Gestión general de las funciones .....	20
3.3.2 Gestión de la energía .....	22
3.3.3 Gestión de los usuarios .....	25
3.3.4 Gestión de alarmas y peticiones .....	28
<b>CAPITULO 4. ADQUISICIÓN DE DATOS.....</b>	<b>32</b>
4.1 Sistema de adquisición de las variables meteorológicas.....	32
4.1.1 Adquisición y adecuación de señal del Hidrómetro.....	32
4.1.2 Adquisición y adecuación de señal del Barómetro.....	33
4.1.3 Adquisición y adecuación de señal del Termómetro.....	34
4.1.4 Adquisición y adecuación de señal del Luxómetro .....	35
<b>CAPITULO 5. ALIMENTACIÓN DE LA ESTACIÓN METEOROLÓGICA .....</b>	<b>36</b>
5.1 Alimentación de la placa.....	36
5.2 Elección de la batería.....	37
5.3 Elección del panel solar.....	38
<b>CAPITULO 6. SISTEMA DE TRANSMISIÓN DE DATOS.....</b>	<b>41</b>
6.1 Características del MODEM GSM.....	41
6.2 Gestión del MODEM GSM .....	43

6.3	Conexión con la estación meteorológica .....	43
<b>CAPITULO 7. DISEÑO DE LA PLACA .....</b>		<b>45</b>
7.1	Esquema general.....	45
7.2	Diseño del circuito impreso. ....	47
7.3	Aspecto terminado del circuito.....	49
<b>CONCLUSIONES .....</b>		<b>50</b>
<b>BIBLIOGRAFÍA .....</b>		<b>51</b>
<b>ANEXOS .....</b>		<b>52</b>
10	FUNCIONAMIENTO IC2 Y MPLAB.....	53
11	PROGRAMACIÓN ESTACIÓN METEOROLÓGICA EN C .....	59
12	CONFIGURACIÓN Y USO DEL HYPER TERMINAL.....	70

## CAPÍTULO 1. INTRODUCCIÓN

### 1.1. Objetivo

El objetivo del proyecto es el diseño de una estación meteorológica autónoma. Dicha estación es un elemento útil para poder tomar mediciones de diversas variables meteorológicas, sin necesidad de desplazarnos al lugar de emplazamiento de la estación.

Podrá ser emplazada en un campo para controlar de forma sencilla por un agricultor los efectos del clima en sus cosechas en un momento determinado, a fin de poder gestionar los recursos a su disposición. Otra aplicación sería la ubicación de la estación en una montaña o bosque, a fin de controlar posibles riesgos de incendios o como consulta para excursionistas antes de desplazarse al lugar.

Las medidas meteorológicas han sido elegidas por su fácil interpretación por un usuario, a fin de detectar posibles lluvias, niebla, heladas, riesgo de incendios, etc. Dichos parámetros y rangos de medición son:

- Temperatura (-20/45 °C)
- Presión (0/2000 milibares)
- Humedad (0/100%)
- Iluminación (0/99999 Lux)

La autonomía de este tipo de dispositivos es un requisito interesante si se desean adquirir las variables meteorológicas en ciertos emplazamientos donde no existen líneas de comunicaciones ni redes eléctricas de alimentación.

Las comunicaciones del sistema se basaran en la red inalámbrica GSM de comunicaciones de teléfonos móviles. Esta red tiene una cobertura en el territorio español del 98 % y por tanto, puede ser utilizada en prácticamente en cualquier lugar.

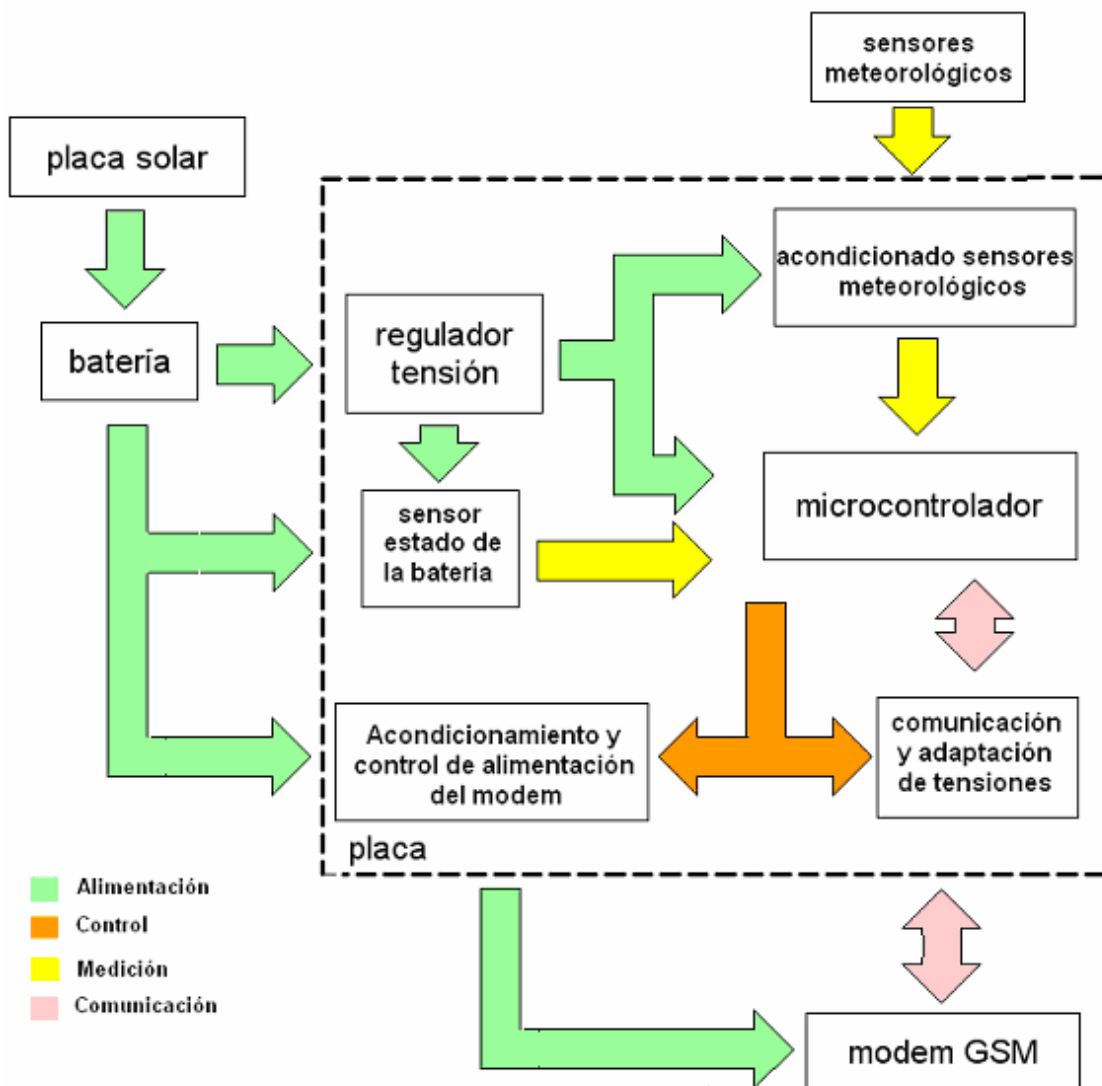
Se pretende alimentar la estación meteorológica a partir de energía solar mediante paneles solares y baterías. Obviamente, este tipo de alimentación condiciona el diseño de todo el sistema, ya que el consumo del circuito electrónico debe ser el mínimo posible.

El sistema debe ser capaz de dar de alta/baja a los usuarios, atender las peticiones de alguno de los parámetros y habilitar alarmas. Las alarmas son los mensajes sms enviados por el sistema al usuario en el caso de sobrepasar un rango de valores predefinidos en las mediciones. Dicha característica es importante a fin de poder informar al usuario de una posible helada, lluvia, excesivo calor o iluminación.

## 1.2. Planteamiento del proyecto

A continuación se muestra un diagrama de bloques donde se enumeran los elementos que componen el sistema de la estación meteorológica. Sigue el esquema típico de un Sistema de Adquisición y Distribución de señales basado en un microcontrolador.

Consta de cuatro partes diferenciables, dependiendo de la función que realizan en el sistema.



**Fig. 1.1** Diagrama de bloques de la estación meteorológica

Descripción breve de los bloques:

### **Sistema de alimentación.**

**Batería:** La batería es el elemento de almacenamiento de energía necesario para alimentar el sistema cuando las condiciones de iluminación no sean suficientes para obtener potencia del panel solar.

**Placa solar:** Es el elemento utilizado para captar energía del ambiente proporcionando al sistema autonomía en su alimentación. Será elegida en función de las necesidades de consumo del circuito y teniendo en cuenta la ubicación de la estación meteorológica, ya que es importante para el cálculo de su dimensionado.

**Regulador de tensión:** Es necesario para acondicionar la tensión de la batería y la placa solar a los niveles requeridos para los dispositivos electrónicos que integran el sistema. El circuito será alimentado íntegramente a 5V, ya que los componentes serán seleccionados para funcionar en dicho rango de tensiones.

### **Sistema de adquisición y acondicionado de señales.**

**Sensores meteorológicos y acondicionamiento:** Se compondrá de 4 sensores con sus respectivos circuitos de acondicionamiento y amplificación mediante un LM324N. Dichos sensores medirán: temperatura, humedad, presión atmosférica y luz. Por otra parte, se han utilizado circuitos acondicionadores para adaptar el rango de medida de los sensores a las tensiones de referencia del conversor analógico digital.

**Estado de la batería:** El estado de la carga de la batería es de gran importancia debido a que el MODEM no funciona con tensiones inferiores a 8V. Mediante un circuito se comprueba su estado y dependiendo de la carga de batería, varía el intervalo de tiempo de desconexión del MODEM.

### **Unidad central de control del sistema.**

**Microcontrolador:** Constituye el centro de procesamiento de los datos enviados por los sensores y el resto de elementos que integran el proyecto. Dispone de puertos de entrada y salida mediante los que controla la alimentación del MODEM y del módulo de comunicación de RS-232.

### **Sistema de comunicaciones.**

**Acondicionamiento y comunicación:** Adapta los niveles de tensión del microcontrolador a los niveles típicos en una comunicación RS232.

**MODEM GSM:** El MODEM GSM permite la comunicación entre el sistema y el usuario vía sms. Se comunica con el sistema mediante el Standard rs232.



## CAPITULO 2. Unidad central de control

### 2.1 Introducción a los microcontroladores

Existen diferentes alternativas para implementar este bloque: máquinas de estado mediante elementos discretos digitales como TTLs, microcontroladores, microprocesadores, DSPs, FPGAs, ASICs, PsoC, etc....

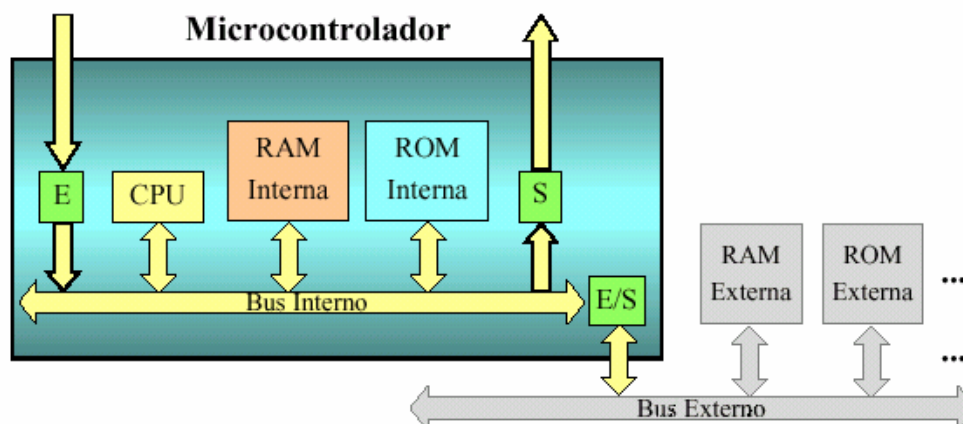
La elección de microprocesadores para dicho proyecto es debido principalmente a dos factores:

- Son dispositivos programables que permiten programar unas funcionalidades específicas para una determinada aplicación.
- Integran en un chip no únicamente una unidad central de proceso sino también todas aquellas funcionalidades típicas de un controlador. Entre estas funcionalidades cabe destacar los puertos de entrada y salida digitales, los analógicos, puertos de comunicaciones, etc.

Un microcontrolador es un circuito integrado programable que contiene todos los componentes de un computador aunque de limitadas prestaciones. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de controlador incrustado (*Embedded controller*).

En su memoria sólo reside un programa destinado a gobernar una aplicación determinada. Sus puertos de entrada/salida soportan el conexionado de sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gestionar la tarea asignada.

Según el tipo empleado pueden diferenciarse en la capacidad y tipo de memoria, en el número de entradas/salidas, cantidad de temporizadores y de convertidores A/D y D/A.



**Fig. 2.1:** Estructura típica de un microcontrolador.

## 2.2 Elección del microcontrolador

A la hora de escoger el microcontrolador a emplear en un diseño concreto hay que tener en cuenta multitud de factores: la documentación y herramientas de desarrollo disponibles, su precio, la cantidad de fabricantes que lo producen y por supuesto, las características del microcontrolador (tipo de memoria de programa, número de temporizadores, interrupciones, etc.).

La elección de los microcontroladores de Microchip frente a otros, se debe a características como su bajo precio, velocidad, reducido consumo, pequeño tamaño, facilidad de uso, fácil programabilidad o la abundancia de información y de herramientas económicas de soporte.

El fabricante Microchip realiza una clasificación en gamas. Cada gama tiene unas características en común y que la distinguen de las demás.

### La gama baja.

La gama baja de los PIC encuadra nueve modelos fundamentales en la actualidad. La memoria de programa puede contener 512 bytes, 1 k o 2 k con una longitud de palabra de 12 bits, y ser de tipo ROM, OTP o EEPROM. La memoria de datos SRAM puede tener una capacidad comprendida entre 25 y 73 bytes. Sólo disponen de un temporizador (TMR0), un repertorio de 33 instrucciones y un número de terminales para soportar las E/S comprendido entre 12 y 20. Al no disponer de interrupciones, la pila sólo tiene dos niveles de profundidad.

La tensión de alimentación admite un valor muy flexible comprendido entre 2 y 6,25 voltios, lo cual posibilita el funcionamiento mediante pilas corrientes teniendo en cuenta su bajo consumo, menos de 2 mA a 5 V y 4 MHz

### La gama media.

En esta gama sus componentes añaden nuevas prestaciones a las que poseen los de la gama baja, haciéndolos más adecuados en las aplicaciones complejas. El repertorio es de 35 instrucciones y la longitud de las instrucciones es de 14 bits. Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertas serie y diversos temporizadores.

Algunos modelos disponen de una memoria de instrucciones del tipo OTP y otros de memoria EEPROM.

El temporizador TMR1 del que dispone esta gama tiene un circuito oscilador que puede trabajar asíncronamente y que puede incrementarse aunque el microcontrolador se halle en el modo de reposo, posibilitando la implementación de un reloj en tiempo real.

Las líneas de E/S del puerto B presentan unas resistencias de carga *pull-up* activadas por software.

## La gama alta.

En la actualidad, esta gama está formada principalmente por tres modelos cuyas características responden a microcontroladores de arquitectura abierta, siendo capaces de ampliar su configuración interna, añadiendo nuevos dispositivos de memoria y de E/S externas. Esta característica obliga a estos componentes a tener un elevado número de terminales comprendidos entre 40 y 44.

Admiten interrupciones, poseen puerto serie, varios temporizadores y mayor capacidad de memoria, que alcanza los 8k palabras en la memoria de instrucciones y 454 bytes en la memoria de datos. El formato de las instrucciones es de 16 bits. El repertorio es de 55 o 58 instrucciones según modelo. La frecuencia máxima de funcionamiento es de 25Mhz, con un ciclo de instrucción de 160ns.

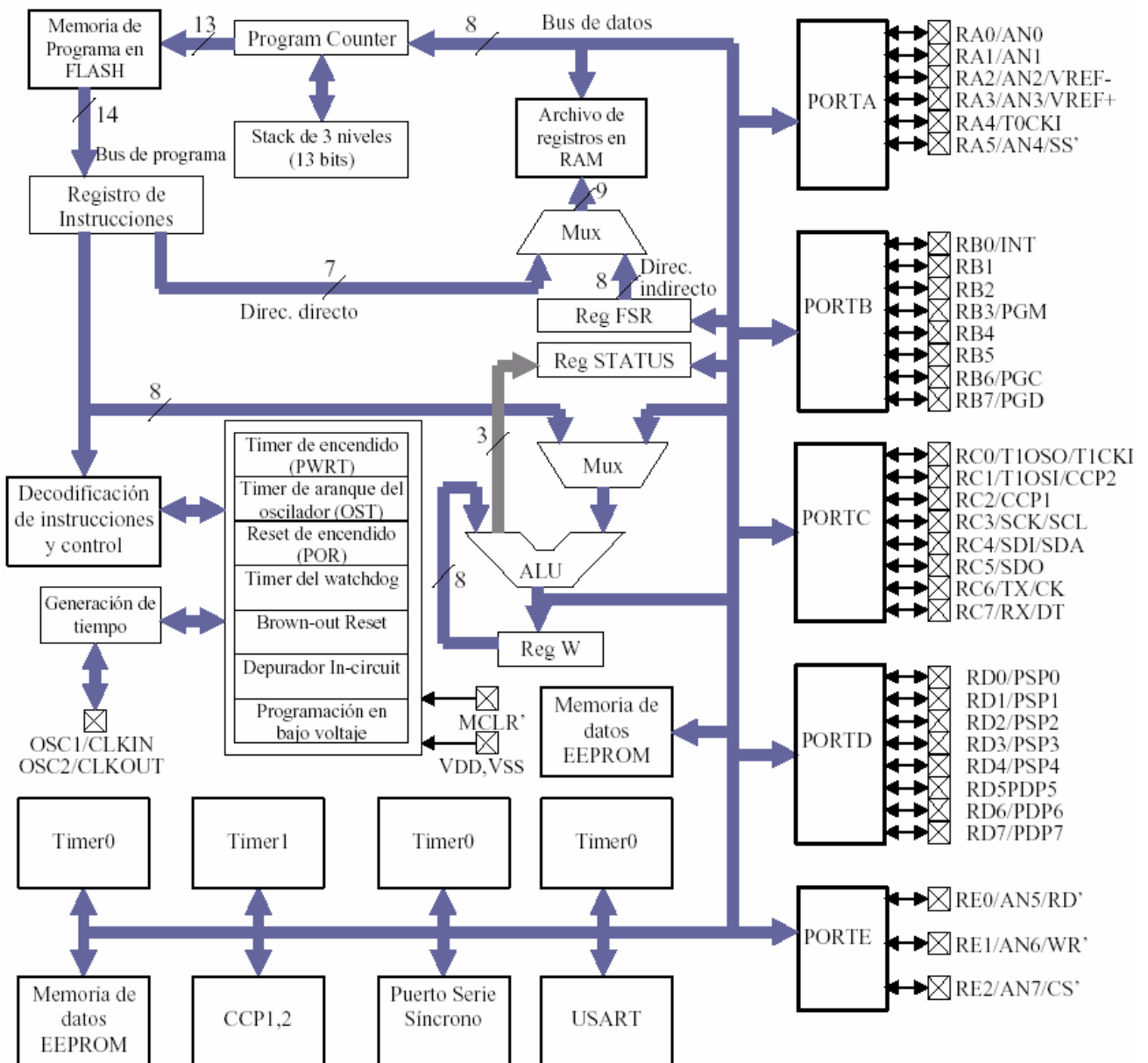
Para la realización de este proyecto es suficiente con utilizar un microcontrolador de la gama media que integre una memoria EEPROM y capacidad de transmisión serie de datos al módem GSM. Debe destacarse que la memoria EEPROM es necesaria para almacenar datos relevantes, tales como teléfonos de usuarios, que no deban perderse en el caso de fallo de la alimentación.

Entre los microcontroladores de gama media se ha seleccionado el 16F877A I/P LF por disponer de mayor capacidad de memoria flash, menor consumo, más posiciones EEPROM y RAM de datos que el resto de microcontroladores de la serie 16FXXX, véase tabla 2.2.

**Tabla 2.2** Comparativa gama 16FXXX

<b>Características</b>	<b>16F873</b>	<b>16F874</b>	<b>16F876</b>	<b>16F877</b>
Frecuencia Máxima	DC-20Mhz	DX-20Mhz	DX-20Mhz	<b>DX-20Mhz</b>
Memoria de programa FLASH Palabra de 14 bits	4KB	4KB	8KB	<b>8KB</b>
Posiciones RAM de datos	192	192	368	<b>368</b>
Posiciones EEPROM de datos	128	128	256	<b>256</b>
Ports E/S	A, B y C	A, B, C y D	A, B y C	<b>A, B, C y D</b>
Nº de Pines	28	40	28	<b>40</b>
Interrupciones	13	14	13	<b>14</b>
Timers	3	3	3	<b>3</b>
Módulos CCP	2	2	2	<b>2</b>
Comunicaciones Serie	MSSP, USART	MSSP, USART	MSSP, USART	<b>MSSP, USART</b>
entrada en Convertidor A/D	5	8	5	<b>8</b>
Juego de Instrucciones	35 instruc	35 instruc	35 instruc	<b>35 instruc</b>
Longitud de la instrucción	14 bits	14 bits	14 bits	<b>14 bits</b>

En la **Fig. 2.2**, se muestra un diagrama de bloques con la organización interna del PIC16F877A I/P LF, que permite tener una visión de la arquitectura de este microcontrolador.



**Fig. 2.2** Diagrama interno de un PIC 16F877A

## 2.2 Programación mediante MPLAB

MPLAB es una herramienta para escribir y desarrollar código en lenguaje ensamblador para los microcontroladores PIC. El MPLAB incorpora todas las herramientas necesarias para la realización de cualquier proyecto, ya que además de un editor de textos, cuenta con un simulador en el cual se puede ejecutar el código paso a paso para ver así su evolución y el estado en el que se encuentran sus registros en cada momento.

MPLAB es un software gratuito que se encuentra disponible en la página web de Microchip.

MPLAB permite realizar las siguientes tareas:

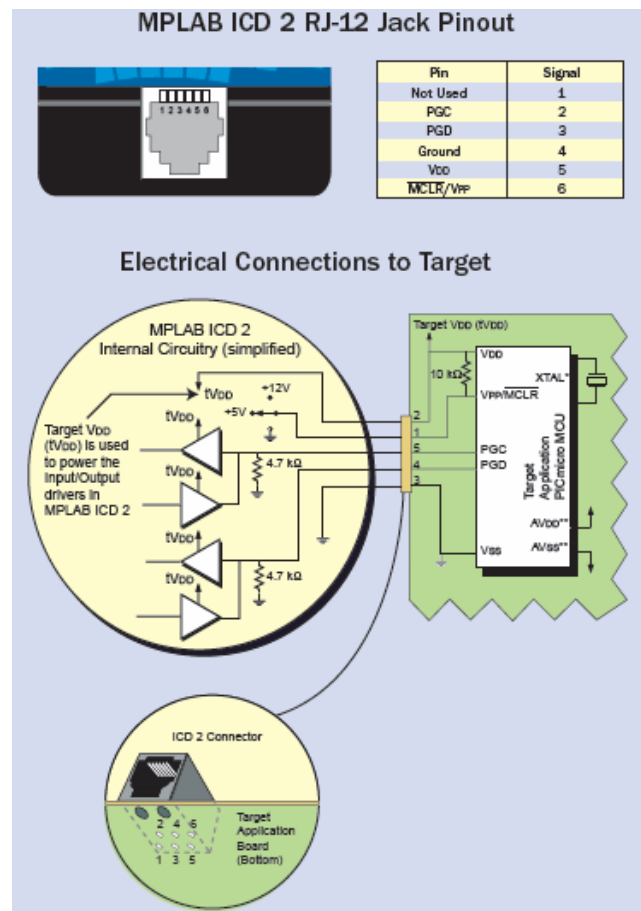
- Manejar el escritorio MPLAB.
- Crear un nuevo archivo de código fuente.
- Identificar y corregir errores simples.
- Ejecutar el simulador interno.
- Marcar puntos de interrupción.
- Crear ventanas de observación.
- Manejar ventanas para el seguimiento de errores.

El programa MPLAB posee una aplicación que permite la grabación del programa en el microcontrolador, siendo necesario en tal caso, disponer de un programador. En este proyecto en concreto se ha empleado el ICD2, debido a que permite tanto la grabación, como la ejecución del programa paso a paso, controlando siempre el cambio de estado de las variables. Una vez instalado el programa MPLAB, se instala el programador ICD2, mostrado en la **Fig. 2.3**



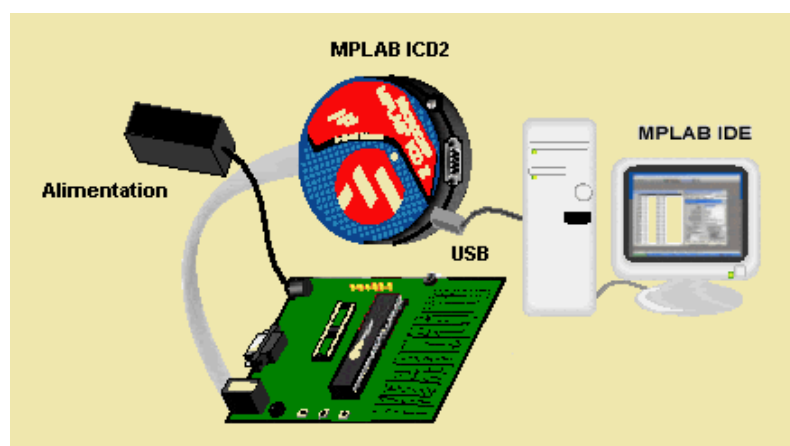
**Fig. 2.3** Programador ICD 2

A la hora de realizar la placa del circuito se tiene en cuenta el utilizar un conector rj-12 como se especifica en la **Fig. 2.4**



**Fig. 2.4** Conexionado ICD2 a la placa

Quedando el esquema general de conexionado, véase **Fig. 2.5**



**Fig. 2.5** Esquema general conexionado ICD2

## CAPITULO 3. Desarrollo del programa.

En este capítulo se describen la funciones del microcontrolador en el sistema y como se han implementado mediante las rutinas programadas.

### 3.1 Funciones del microcontrolador en el sistema

La función que realiza el microcontrolador en la estación meteorológica se agrupa en cinco bloques principales, los cuales se componen de diversas funciones que se irán desglosando en este capítulo.

- Gestión de los usuarios: Altas y bajas de los usuarios.
- Gestión de las alarmas y peticiones: Respuesta a los mensajes y activación de las alarmas.
- Gestión de los datos: Comprobación del estado de la batería y sensores meteorológicos.
- Gestión de la energía: Encendido y apagado del MODEM y de más periféricos.
- Gestión general de las funciones: Relacionar las diferentes funciones de forma coherente para su funcionamiento.

### 3.2 Definición de las entradas y salidas del PIC

A la hora de programar el PIC, se ha definido el uso de las diferentes entradas y salidas, véase **Fig. 3.1**. El PIC utilizado consta de siete puertos de entrada que pueden ser configurados para la captación de señales analógicas. En este proyecto, se emplean únicamente cinco de ellos para las medidas de los sensores y del estado de la batería.

AN0- Entrada analógica sensor luminancia  
AN1- Entrada analógica sensor temperatura  
AN2- Entrada analógica sensor presión  
AN3- Entrada analógica sensor humedad  
AN7- Entrada analógica estado de la batería

Otros puertos a tener en consideración son los habilitados para la comunicación RS-232. Para ello el PIC tiene predefinidos dos puertos, los cuales se habilitan para tal efecto mediante el programa (RC6-Transmisión RS-232, RC7-Recepción RS-232).

Debido a que el diseño del sistema se ha orientado para tener mínimo consumo de energía, mediante el PIC se habilitan los distintos elementos que forman la estación meteorológica. A través del PIC se controla la alimentación de los sensores, su acondicionamiento de señal (RDO) y la medición del estado de la batería (RD1), así como la conexión del MODEM (RD2) y la conexión del chip de comunicaciones RS-232 (RC5).

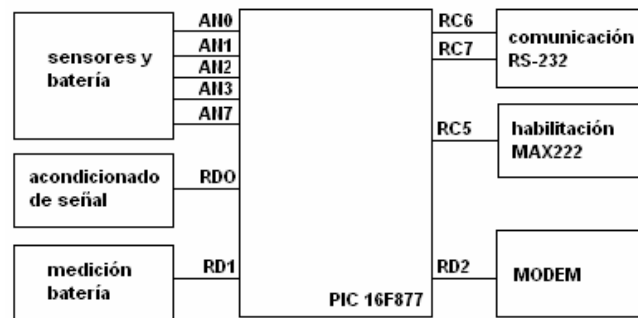


Fig. 3.1 Conexión del PIC

### 3.3 La programación de la estación meteorológica

El programa principal se encarga de configurar el microcontrolador y llamar a las diferentes funciones según que tarea deba realizar el PIC en cada momento. También, como parte de la estructura del programa principal deben incluirse 2 interrupciones (Interrupción del *Timer* y la Interrupción de la recepción de datos).

La Figura 3.2 muestra la jerarquía de las diferentes funciones que se han utilizado y que serán posteriormente explicadas en detalle.

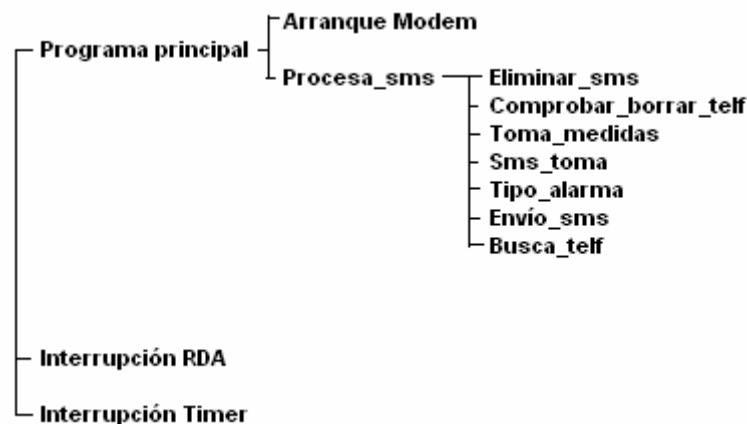


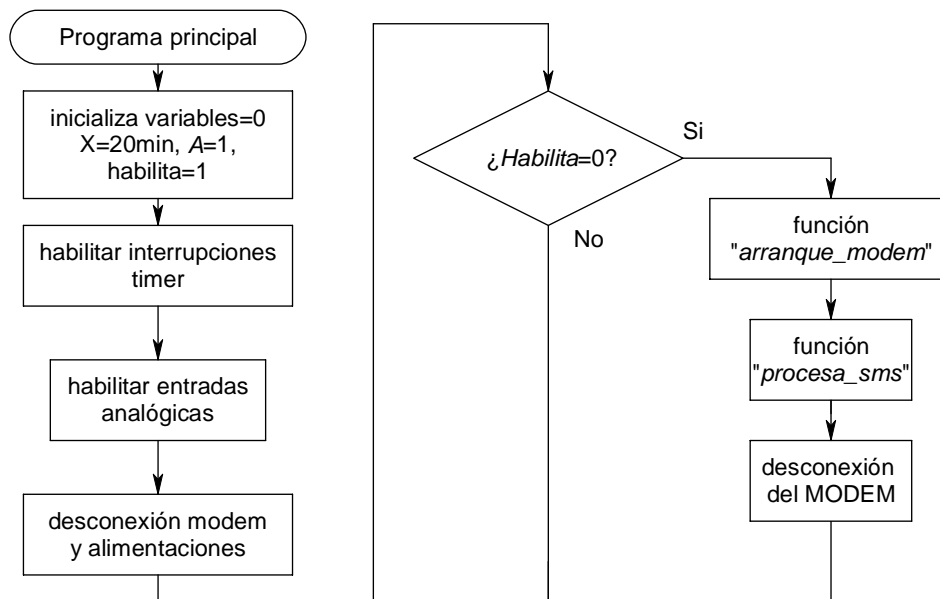
Fig.3.2 Estructura del programa

En el programa principal, véase **Fig. 3.3**, se habilitan todos los parámetros del PIC para su correcto funcionamiento (inicializaciones de variables, definiciones de entradas y salidas, configuración del timer). La variable A se empleará como indicador de que posición de mensaje se ha de leer del MODEM.

Una vez inicializado el sistema, el programa principal entra en una rutina de espera de veinte minutos. Transcurrido ese tiempo se encenderá el MODEM y procederá a procesar los mensajes que se hayan transmitido desde los usuarios. Finalmente, se apagará el MODEM y se esperará otro intervalo de tiempo hasta la próxima conexión. El tiempo entre conexiones viene determinado por la variable X y depende del estado de la batería.



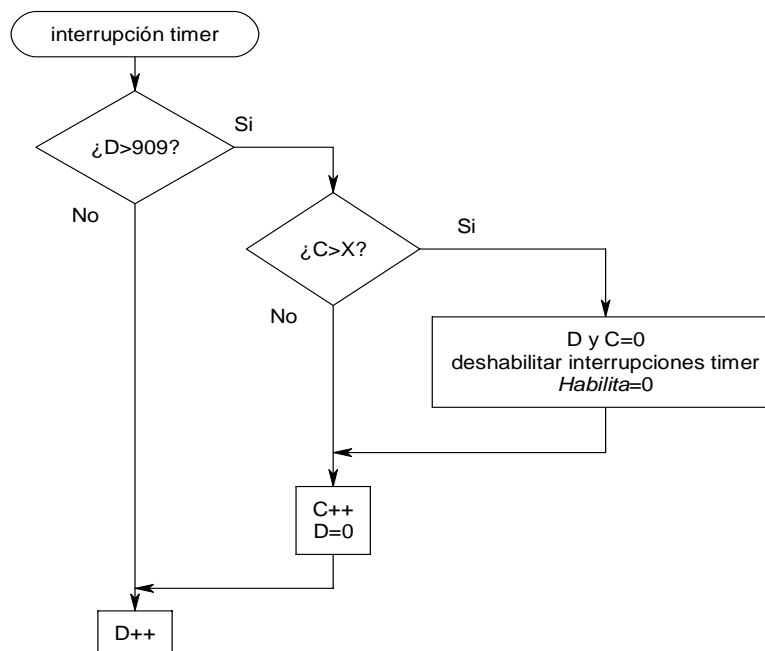
La temporización del tiempo entre conexiones se realiza mediante una interrupción activada por el timer 1. Cuando ha transcurrido el tiempo indicado por X, la interrupción pone la variable *Habilita* a cero indicando al programa principal que ya ha transcurrido el tiempo.



**Fig. 3.3** Programa principal

### Interrupción del temporizador

La interrupción provocada por el Timer 1, véase **Fig. 3.4**, tiene por objeto medir los tiempos entre las conexiones del MODEM y así gestionar la energía en la batería.



**Fig. 3.4** Interrupción de temporización

La interrupción gestiona el incremento de dos variables locales, C y D. La llamada a la interrupción se realiza cada 33ms (valor predefinido anteriormente en la configuración del timer). Una vez producida la interrupción se incrementa el valor de la variable D.

Cuando se repita la operación 909 veces, es decir transcurridos 30 seg., se incrementará en una unidad la variable C. La variable C se incrementará hasta que alcance el valor límite de temporización determinado por la variable X.

$$\begin{aligned} T &= 33ms * 909 = 30seg \\ T_{total} &= T * valor\_variableX \end{aligned} \quad (3.1)$$

Finalizado el proceso se desactiva la interrupción y se pone a cero la variable *Habilita*, indicando así al programa principal que puede proceder con una nueva conexión del modem.

#### Interrupción para la recepción de datos

Dicha interrupción se produce al recibir algún carácter proveniente del MODEM vía RS-232 (interrupción RDA, recepción de datos serie), véase **Fig. 3.5**. Al conectar el MODEM se habilita la recepción de datos mediante esta interrupción. Comprueba si hay información en los mensajes y si la información es correcta.

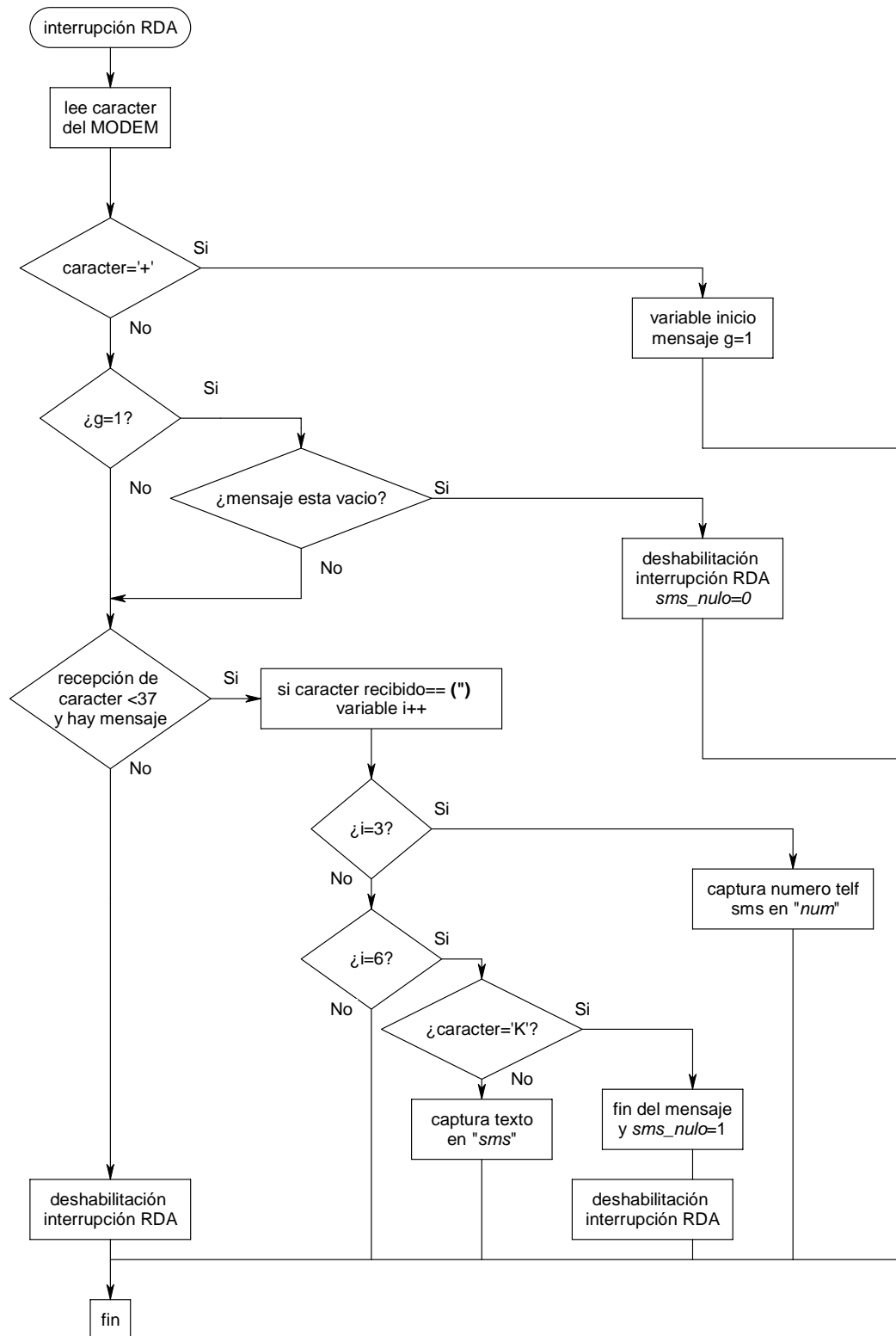
Del mensaje sólo se guarda la información necesaria, es decir, el número de teléfono y los datos relevantes del sms. Esto se debe a que el PIC tiene un espacio limitado para guardar datos en la RAM y así se consigue minimizar el espacio ocupado en memoria.

Para seleccionar los datos del mensaje que son de interés se tiene en cuenta la estructura de los mensajes. Los datos se reciben carácter a carácter. Mediante la variable G, se controla el inicio del mensaje y capturan los caracteres a partir de las terceras comillas, donde se encuentra el número de teléfono del usuario que lo ha enviado. Dicha operación se repite a partir de las comillas número seis donde se encuentra el texto del sms. Véase el siguiente ejemplo.

**+CMGL: 1,"REC READ","+34609287222",,"07/07/26,23:48:49+08"texto OK**

Una vez terminada la recepción del sms, la función devuelve la variable *sms\_nulo* indicando si hay mensaje o no, y si hubiera información se guarda el número de teléfono del usuario en la cadena de caracteres *num* y el mensaje en la cadena de caracteres *sms*.

Una vez finalizado el sms se recibe un 'OK' que indica el final de mensaje. Se deshabilita la recepción de sms (deshabilita interrupción RDA), ya que se ha de procesar la información recibida.

**Fig. 3.5** Interrupción de recepción de datos

### 3.3.1 Gestión general de las funciones

#### Función “*procesa sms*”

Una vez encendido el MODEM y obtenido el mensaje mediante la interrupción RDA, la función *procesa\_sms*, véase la **Fig. 3.6**, se encarga de interpretar el mensaje y vigilar las alarmas.

Dicha función se ejecuta después de la función “*arranque\_modem*”. Cabe destacar, que si se comprueba que han sido recibidos tres sms sin información (la variable B es la encargada de ello) el sistema se desconecta y espera el tiempo asignado en el timer para su próxima activación (activa la interrupción del timer). También, será reiniciada la variable A, cuya función en el programa es indicar el mensaje que se lee del MODEM.

En primer lugar se procede a comprobar el valor de *sms\_nulo* a fin de determinar si hay mensaje o no. Si existe mensaje se procede a la eliminación del sms de la memoria del MODEM, ya que los datos de interés están guardados en la memoria del PIC.

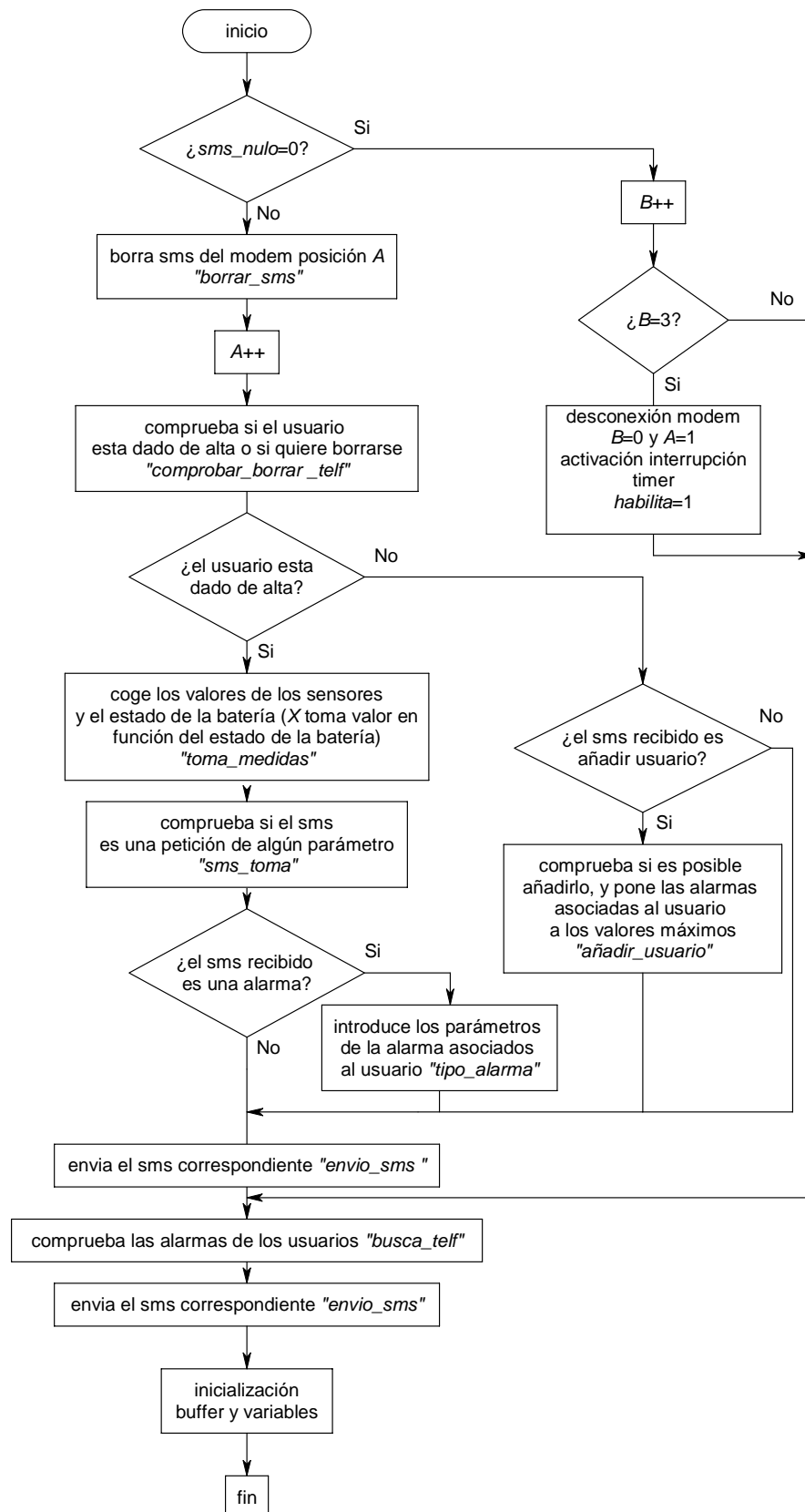
Si el sms contiene información, se realiza la comprobación de que el usuario que lo ha enviado está dado de alta en el sistema. Para ello, se comprobará si el número recibido coincide con alguno de los almacenados en la memoria, y si su petición es ser dado de baja del sistema. Este buscará el número en la memoria y lo eliminará poniendo los valores de las alarmas asociados a los valores de origen.

Si no está dado de alta se comprueba si el sms es una petición de alta de un nuevo usuario, y si queda memoria disponible para darlo de alta. Si no fuera posible se le envía un sms denegando su petición. Una vez dado de alta el nuevo usuario, se establecerán las alarmas para ese usuario a los valores máximos. La palabra clave para darse de alta es “alta”, así pues, también sería posible indicarlo mediante una contraseña o “alta+contraseña”.

Si el usuario está dado de alta se comprueba si ha hecho una petición de alguno de los 4 parámetros o quiere activar alguna alarma. Si es así, le será enviado el sms correspondiente.

Una vez realizadas dichas funciones y enviado dicho sms al usuario, el sistema comprueba si se ha activado alguna alarma de alguno de los usuarios y le envía un sms avisándole de dicha alarma.

El desarrollo de las diferentes funciones se desglosa en los siguientes apartados.

Fig. 3.6 Función *procesa\_sms*

### 3.3.2 Gestión de la energía

El MODEM es el elemento que tiene mayor consumo de energía en el sistema. Por este motivo, no puede permanecer encendido todo el tiempo y debe únicamente activarse durante intervalos de tiempo pequeños, durante los cuales se atienden tanto a la transmisión de mensajes como a su recepción. La frecuencia de activación del MODEM depende del estado de la batería, según se muestra en la tabla.

Carga batería 110mA	Tiempo en espera	Valor variable X
>11V	20 min.	40
11V-10V	40 min.	80
10V-9V	60 min.	120
<9V	120 min.	240

**Tabla 3.1** Valores asignados a X en función de la carga

La medición de los intervalos de tiempo se realiza con la interrupción anteriormente comentada, siendo X la variable de control del temporizador. Dicha variable toma su valor en la función “*toma\_mediciones*”.

### 3.3.3 Gestión del MODEM

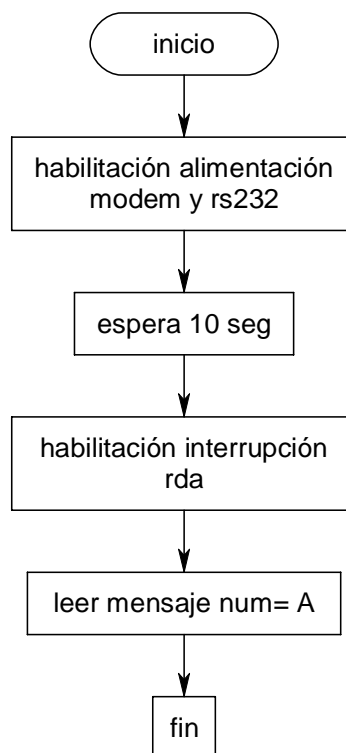
Las funciones que hacen referencia a la gestión de datos son las encargadas de habilitar el MODEM y hacer peticiones de mensajes, procesar los sms, separar la información de interés, eliminar los sms, enviar a los usuarios sms con la información requerida y realizar la captación de las medidas de los sensores.

### La función “arranque MODEM”

Esta habilita el encendido del MODEM y del chip de comunicaciones rs-232, véase la **Fig. 3.7**. Dicha función es llamada desde el programa principal. El MODEM tiene un tiempo de activación de 5 seg. Para poder recibir un sms son necesarios otros 5 seg. para que comience a recibirlos.

Se ha de habilitar la interrupción RDA, dicha interrupción permite la recepción en el PIC de caracteres y se activará cada vez que se reciba un caracter desde el MODEM.

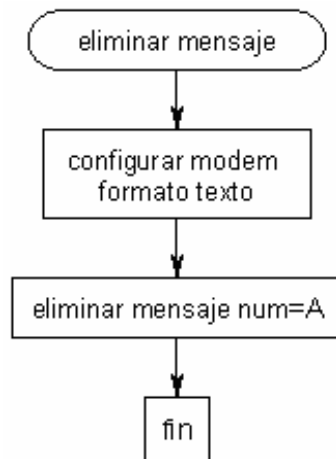
Una vez realizadas dichas operaciones se procede a enviar al MODEM un mensaje para que nos permita leer el primer sms. El mensaje en cuestión será AT+CMGR=A. Dicho valor, variable A, irá incrementándose en sucesivas ocasiones, estando inicialmente a valor 1, a fin de leer el primer sms del MODEM.



**Fig. 3.7** Función *arranque\_MODEM*

### Función “eliminar mensaje”

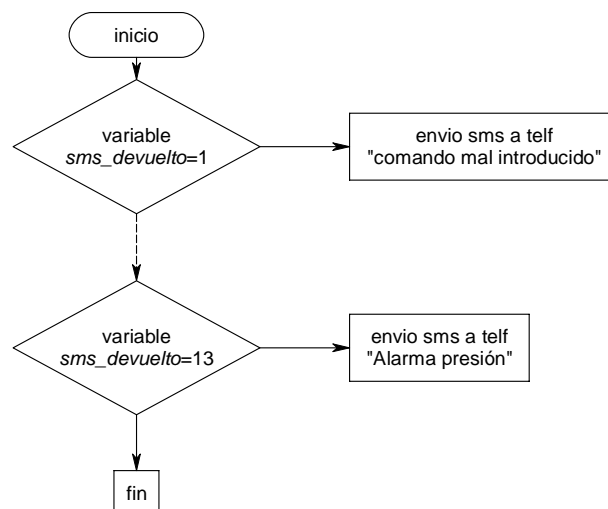
Una vez se ha guardado en el PIC la información relevante del mensaje, se procede a su eliminación en la memoria del MODEM, véase **Fig. 3.8**. Esta operación se realiza mediante el comando AT+CMGD=A, siendo A el número de posición del mensaje que se desea eliminar. Dicho valor irá incrementado para poder eliminar los sms sucesivos.



**Fig. 3.8** Función *eliminar\_sms*

### Función “envío sms”

Esta es la encargada de enviar el sms correspondiente al usuario cuando este realiza una petición o bien, se activa una alarma, véase la **Fig. 3.9**. Esta función tiene como variable de entrada *sms\_devuelto* la cual le indica que tipo de mensaje debe enviar. El valor de *sms\_devuelto* viene determinado en las funciones posteriores. Por simplicidad para la interpretación de los flujogramas se indicará el mensaje que será enviado. No obstante, realmente toma un valor que en esta función interpreta que sms ha de enviar.



**Fig. 3.9** Función envío sms



### 3.3.3 Gestión de los usuarios

Debido a la reducida capacidad de memoria de este PIC se ha limitado a dos posibles usuarios la estación meteorológica. Estos usuarios se identificarán por su número de teléfono, la respuesta será enviada a dicho número.

Cada uno de los usuarios podrá acceder al sistema mediante mensajes sms y pedir información de las variables meteorológicas, definir alarmas de aviso, o incluso gestionar la baja del propio usuario.

Los mensajes que pueden ser enviados por los usuarios son:

Darse de alta:	Alta
Darse de baja:	Baja
Petición de mediciones:	Temperatura Presión Humedad Luz
Activación de las alarmas:	Alarma.temperatura(max:XXX/min:XXX) Alarma.presion(max:XXXX/min:XXXX) Alarma.humedad(max:XXX/min:XXX) Alarma.luz(max:XXXXXX/min:XXXXXX)

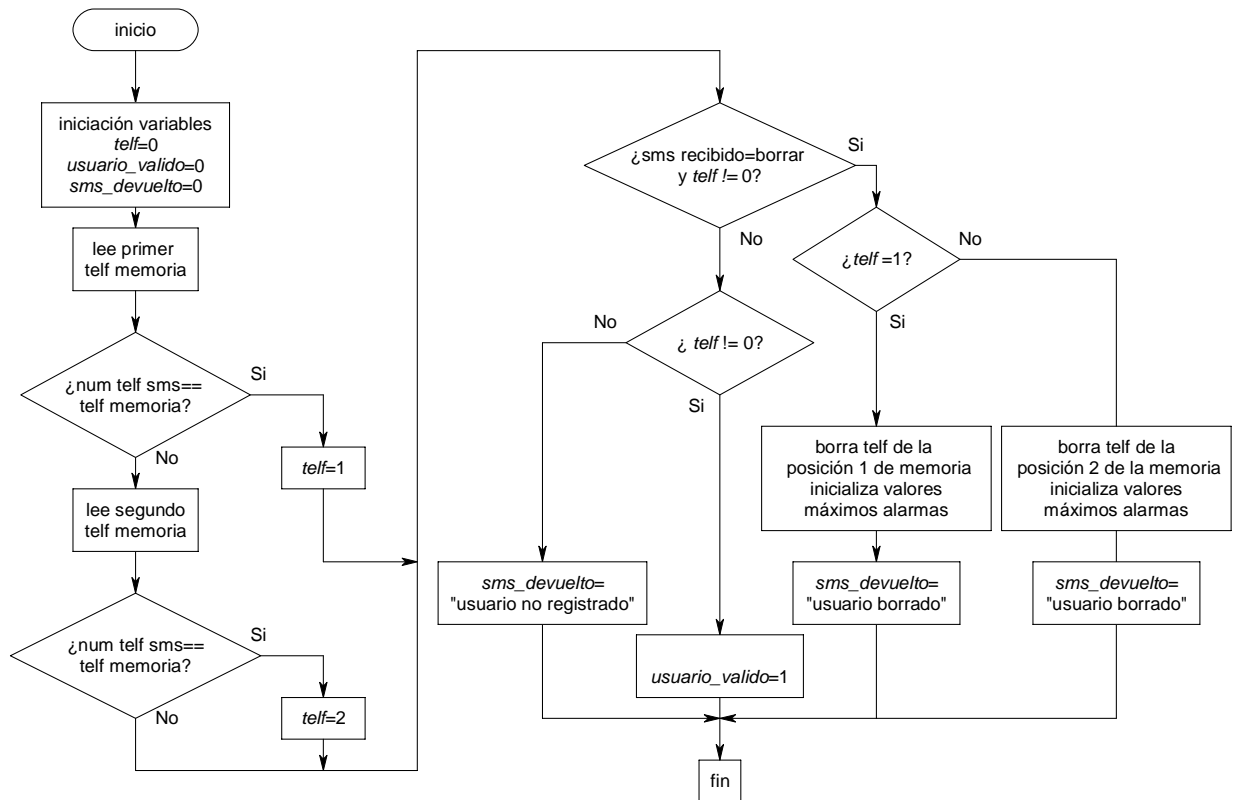
El sistema les responderá mediante los mensajes preestablecidos:

- Comando mal introducido
- Usuario borrado
- Usuario aceptado
- Memoria llena
- Usuario no registrado
- La temperatura es: +/-XXX F
- La presión es: XXX milibares
- La humedad es: XXX %
- La luminosidad es: XXXXX Lux
- Alarma introducida
- Alarma por Temperatura
- Alarma por Humedad
- Alarma por Presión
- Alarma por Luminosidad

### Función “comprobar borrar telf”

Para verificar si un usuario está dado de alta en el sistema, se compara el número de teléfono recibido (variable *num*) con los números de teléfono que tiene el sistema en su memoria, véase la **Fig. 3.10**. Si se encuentra alguna coincidencia la variable *telf* indica la posición de memoria en la que se ha encontrado el teléfono del usuario y se activa la variable *usuario\_valido*.

Para poder darse de baja hay que comprobar que el usuario está dado de alta en el sistema, y su petición sea un sms de baja (comprueba si la variable *sms* es una baja). Si es así, se procede a eliminarlo de la memoria del sistema y deshabilitar las alarmas que tuviera establecidas. También cabe la posibilidad de que un usuario no registrado intente acceder a la estación. En tal caso, le será enviado un sms denegando su petición (variable *sms\_devuelto* toma un valor correspondiente al mensaje “usuario no registrado”).

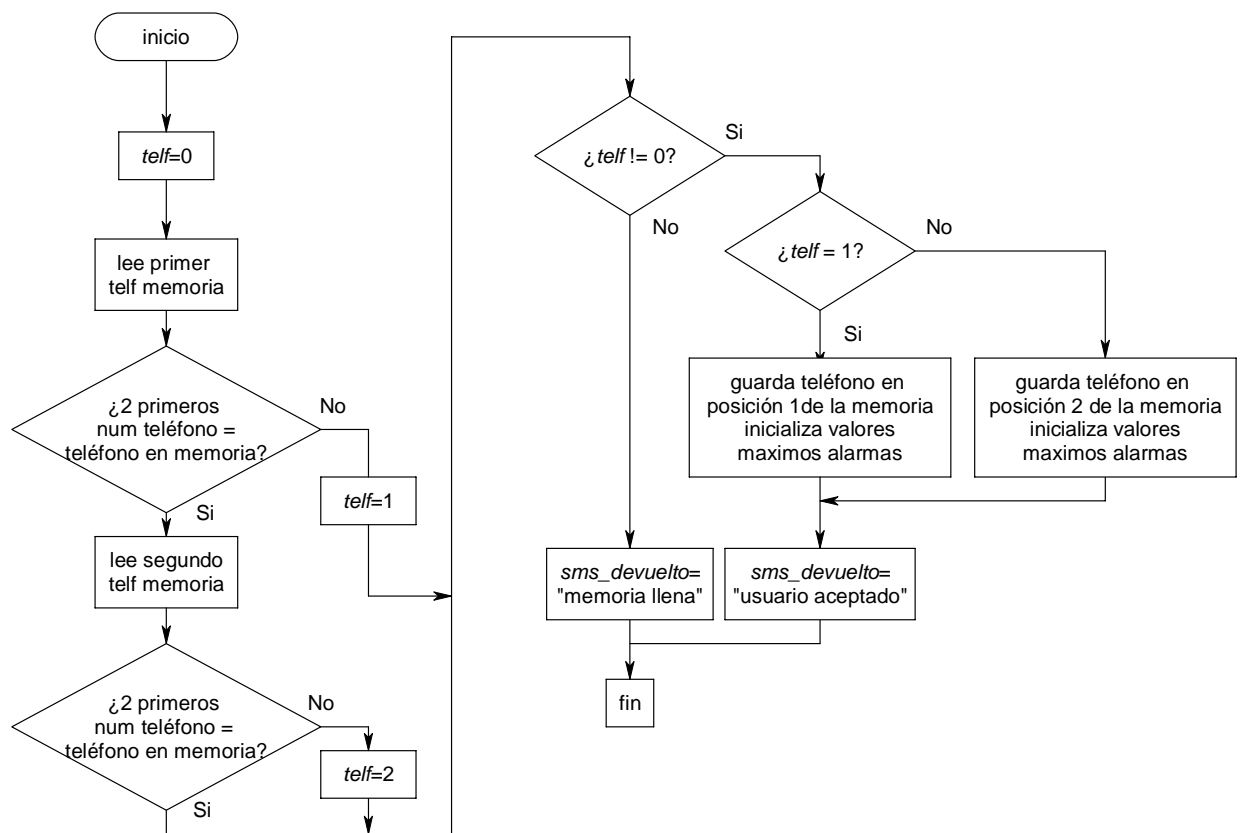


**Fig. 3.10** Función *comprobar\_borrar\_telf*

### Función “añadir telf”

Previamente al envío de un mensaje a un usuario que no está registrado, se comprueba mediante la función *añadir\_telf*, véase **Fig. 3.11**, si su petición es darse de alta. Para ello, una vez comprobado si el sms enviado (variable sms) es una nueva alta, se verifica si existe alguna posición de memoria sin asignar para poder añadirlo.

Una vez localizada se procede al almacenado en memoria del nuevo usuario e inicialización de los valores asociados de las alarmas a valores máximos. Finalmente, acaba la función asignando a la variable *sms\_devuelto* la cadena de caracteres “memoria llena” o “usuario aceptado”.



**Fig. 3.11** Función *añadir\_telf*

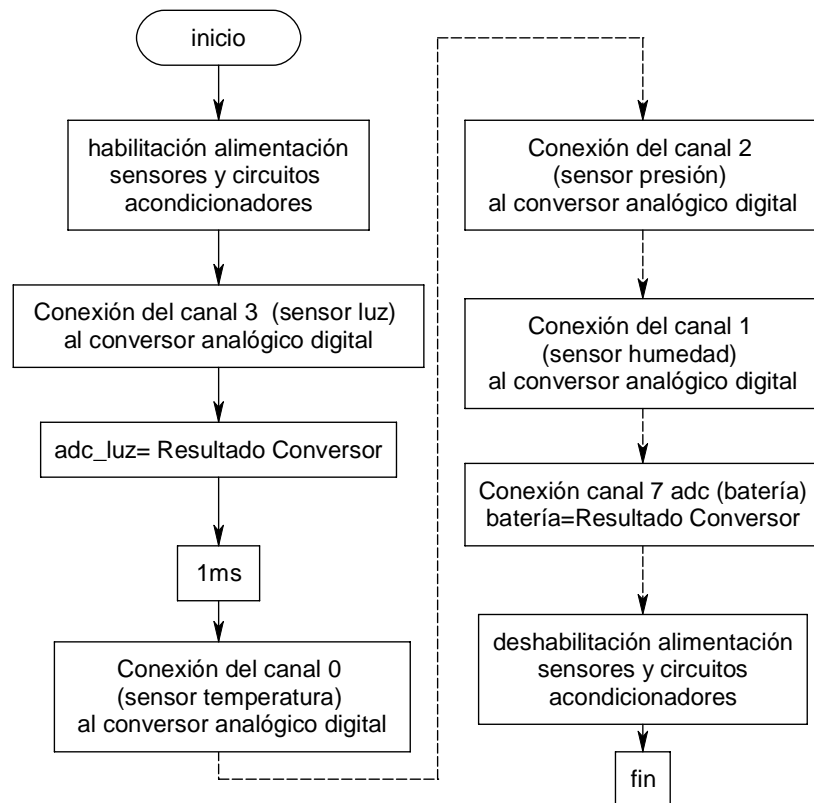
### 3.3.4 Gestión de alarmas y peticiones

El microcontrolador dispone de varias entradas analógicas que son utilizadas para captar la señal de los sensores. Todas estas entradas comparten un mismo convertor analógico/digital que está multiplexado. La toma de medidas se realiza de forma secuencial con un breve intervalo de tiempo entre cada medición (1ms).

#### Función “toma\_medidas”

La captura de las mediciones de los sensores, véase **Fig. 3.12** se realiza mediante la función “toma\_medidas”. Se activa la alimentación de los sensores, se capturan las medidas, y se ajustan a los formatos establecidos para cada tipo de señal. Además se determina el estado de la batería a partir de su tensión para una carga de 100 mA. El nivel de tensión medido determinará el tiempo que debe transcurrir hasta que vuelva a conectarse el modem, véase **tabla 3.1**

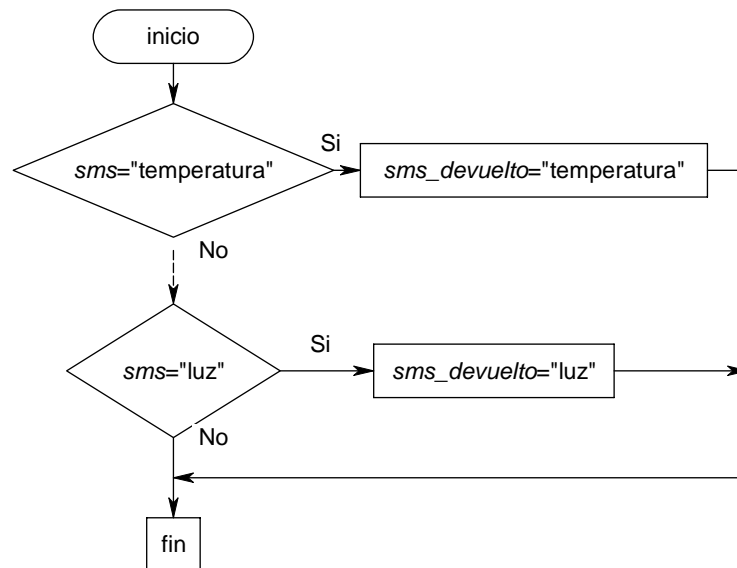
Una vez capturados los valores se deshabilita la alimentación de los sensores y el circuito utilizado para verificar el estado de carga de la batería.



**Fig. 3.12** Función “toma\_medidas”

### La función “*sms toma*”

Comprueba si el usuario ha realizado alguna consulta de las variables meteorológicas, véase **Fig. 3.13**. Compara el mensaje de texto enviado con otros predefinidos como “temperatura”, “presión”, etc... Si encuentra alguna coincidencia envía un sms con la medida solicitada mediante *sms\_devuelto*.

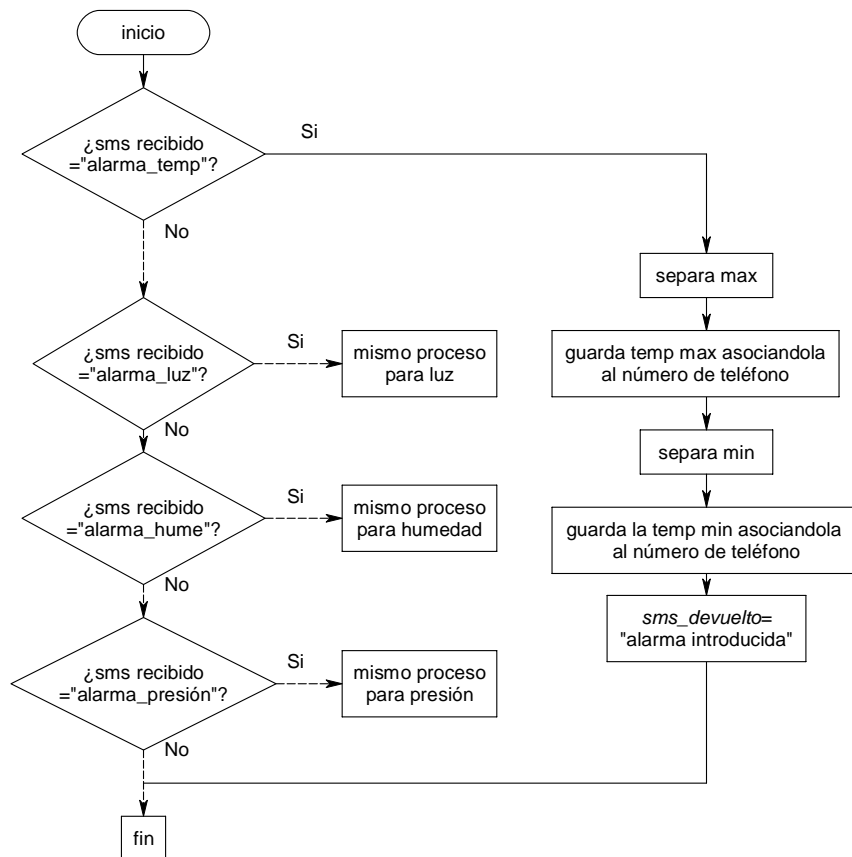


**Fig. 3.13** Función *sms\_toma*

### Función “*tipo alarma*”

Una vez descartada la opción de petición de variables meteorológicas, se comprueba si ha enviado una petición de activación de alguna de las alarmas, véase la **Fig. 3.14**. Mediante mensaje de texto predefinidos se establecen los límites máximos y mínimos a partir de los cuales el sistema deberá enviar un sms de alarma. La función se encargara de separar los valores máximos y mínimos y asociarlos al usuario.

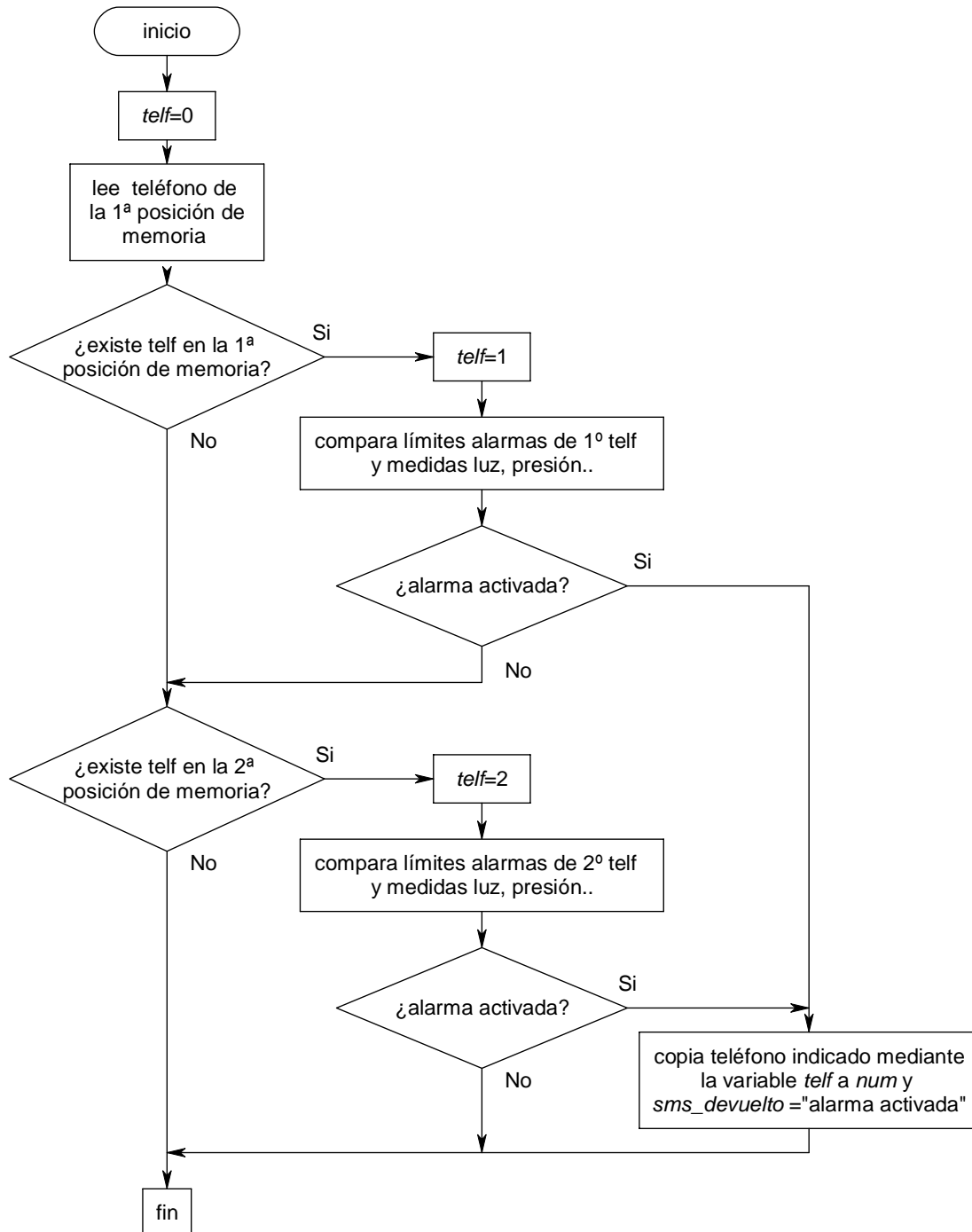
Finalmente, esta función envía un mensaje confirmando que la alarma ha sido introducida correctamente.



**Fig. 3.14** Función *tipo\_alarma*

### Función “busca\_telf”

Para verificar el estado de las alarmas habilitadas por los usuarios, la función “busca\_telf”, véase la **Fig. 3.15**, se encarga de comparar las medidas obtenidas por los sensores con los límites establecidos por cada usuario. En el caso de que estos límites sean superados, envía un mensaje sms indicando el tipo de alarma. Dicho proceso de búsqueda lo realiza mediante la variable *telf*.



**Fig. 3.15** Función *busca\_telf*

## CAPITULO 4. ADQUISICIÓN DE DATOS

### 4.1 Sistema de adquisición de las variables meteorológicas

El sistema de adquisición de las variables meteorológicas está formado por cada uno de los sensores y los circuitos de procesado analógico necesarios para acondicionar adecuadamente estas señales. Por tratarse de una arquitectura de alto nivel se ha utilizado un circuito de acondicionamiento específico para cada sensor que adapta el margen de la mediada a los límites de referencia del conversor analógico digital, en este caso de 0 a 5V.

Se han empleado filtros paso bajo a las salidas en cada uno de los circuitos acondicionadores a fin de disminuir los ruidos o posibles interferencias en cada una de las variables a medir.

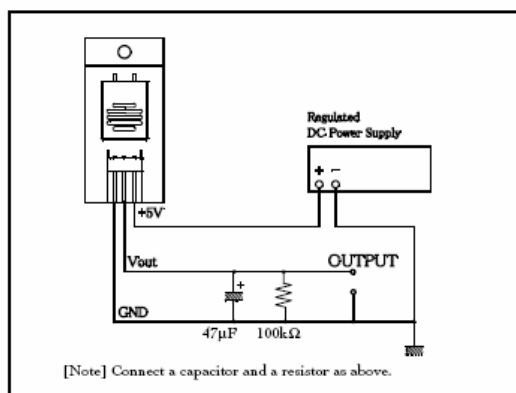
#### 4.1.1 Adquisición y adecuación de señal del Hidrómetro

Para realizar la medida de humedad se ha seleccionado un sensor de humedad relativa compensado HU10 del fabricante Crown Industrial Estate.

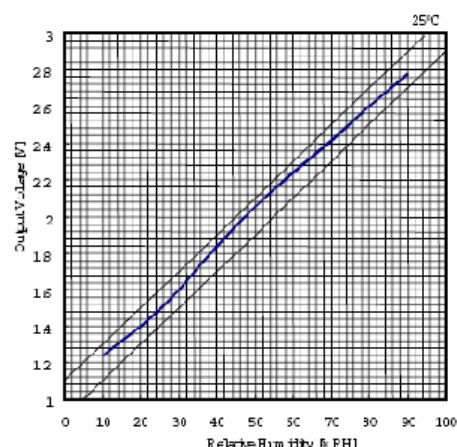
El sensor esta diseñado de fábrica para aplicaciones lineales. Tiene una tensión de alimentación de  $5 \pm 0.2\text{Vdc}$ , y un error de linealidad del  $\pm 5\%RH$  dentro del rango de medidas entre el 25% y el 90%RH. El margen que se desea medir comprende de 0 a 100% de humedad.

El fabricante en la hoja de características ya estipula el esquema que se debe seguir para adecuar la señal al sensor, como denota en la **Fig. 4.1** que viene a continuación. En este caso se comporta como una fuente de intensidad y varía su intensidad de salida en función de la humedad relativa en el ambiente.

#### MEASUREMENT CIRCUIT:



#### OUTPUT CHARACTERISTICS:



**Fig. 4.1** Circuito de adaptación de señal y grafica de V(%RH) a 25°C



Como se puede observar en la gráfica dada por el fabricante, la salida del sensor no es del todo lineal. Para conseguir una relación lineal se realiza una regresión lineal con una serie de datos tomados de la gráfica, véase la **Tabla 4.1**. Para obtener la ecuación de la recta que caracteriza la función, los valores que se han tomado son los siguientes:

Humedad	Volts
0	1.05
10	1.25
20	1.35
30	1.6
40	1.85
50	2.05
60	2.25
70	2.43
80	2.6
90	2.78

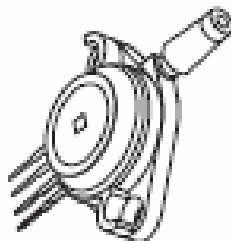
**Tabla 4.1** Datos tomados de la gráfica

La recta que se obtiene siguiendo dicho procedimiento es  $y=0,02x + 1,03$ , con una correlación de 0,99 por lo tanto existe una aproximación y un error total del  $\pm 6\%RH$ .

#### 4.1.2 Adquisición y adecuación de señal del Barómetro

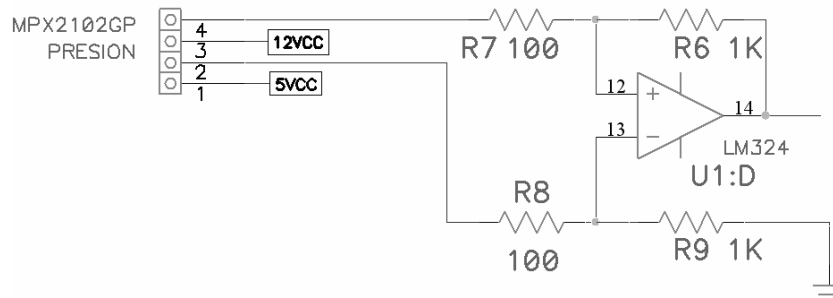
Para realizar la medición se ha escogido un sensor de silicio piezoresistivo de cuatro terminales de la casa Freescale Semiconductors **Fig. 4.2**. La referencia por el que lo denomina el fabricante es MPX2102.

Este sensor tiene una tensión diferencial de salida lineal en relación a la presión atmosférica. El margen de alimentación es de  $\pm 10V$  y  $\pm 16V$ , su sensibilidad es de  $0,4mV/KPa$  y está compensado en temperaturas de 0 a  $85^{\circ}C$ .



**Fig. 4.2** MPX2102

La adaptación de la señal se realiza mediante un amplificador diferencial con una ganancia 10, formado por resistencias de un 1% de tolerancia y un amplificador LM324N, véase la **Fig. 4.3**. Las tensiones diferenciales de alimentación deben ser positivas, en este caso serán 5V y 12V, a fin de elevar el valor de la medición a tensiones positivas, así poder ser amplificadas y adecuadas al rango del microprocesador.



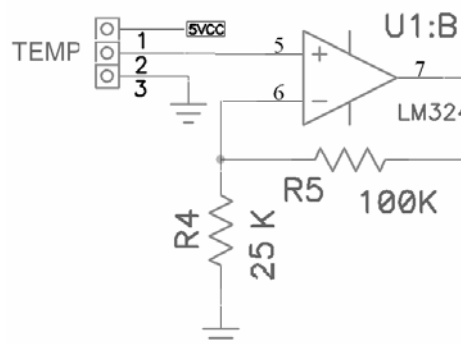
**Fig. 4.3** Circuito de acondicionamiento del sensor de presión

La sensibilidad del sistema con el amplificador incluido queda establecida en 40mV/kPasc de tal forma que el barómetro tiene un fondo de escala de 4V a 100kPasc. También tendrá un error debido al offset del propio sensor y del amplificador. El offset del sensor es de 2mV y una vez amplificado pasa a ser de 20mv. Esto conlleva un error de 0,5kPasc, que después será corregido mediante la programación del PIC.

#### 4.1.3 Adquisición y adecuación de señal del Termómetro

El sensor de temperatura utilizado es un circuito integrado de tres terminales. El TMP36 genera una tensión proporcional a la temperatura absoluta, cumpliendo una relación lineal de 10mV/K. Dicho sensor está específicamente recomendado para temperaturas de -40°C hasta 125°C, la salida a 25°C será de 750mV según las especificaciones el fabricante. El margen de temperaturas que se ha seleccionado es de -40°C a 50°C.

Para acondicionar la señal se ha utilizado otro de los amplificadores operacionales integrados en el LM324N en configuración no inversor **Fig. 4.5**, se establece una ganancia de 5, la sensibilidad de la medida es de 50mV/Kelvin

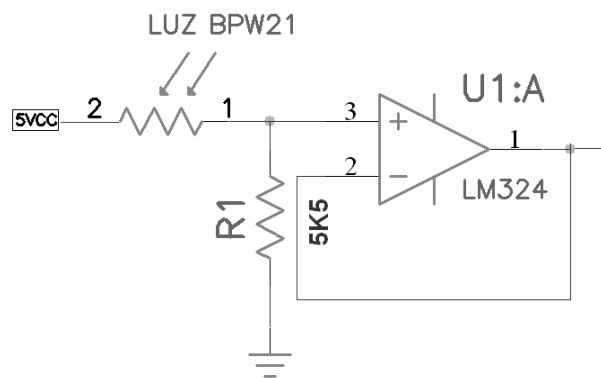


**Fig. 4.5** Circuito de acondicionamiento del sensor de temperatura

#### 4.1.4 Adquisición y adecuación de señal del Luxómetro

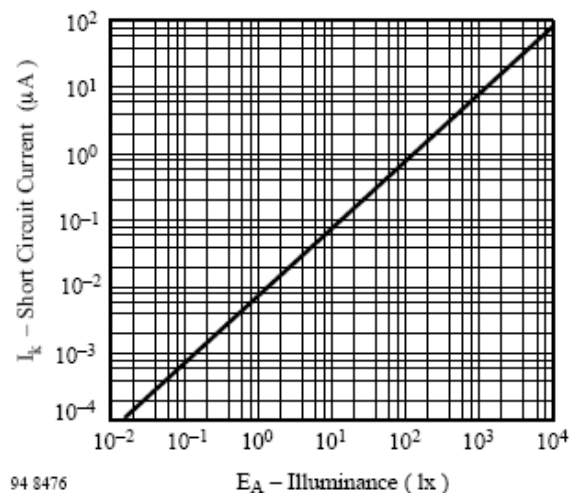
El sensor utilizado es un fotodiodo PN de silicio del fabricante Vishay Telefunken diseñado para aplicaciones lineales. Tiene una ventana de cristal fino diseñada para filtrar unas determinadas longitudes de onda de tal forma que se aproxime lo máximo posible a la respuesta espectral (Visible Light) del ojo humano. El ojo humano percibe de los 280nm a los 780nm.

El esquema eléctrico, véase la **Fig. 4.6**, es un conversor de corriente tensión con la entrada invertida de forma que tendremos a la salida  $V_o = R \cdot I$ . Se ha empleado el LM324N para la amplificación de dicha señal.



**Fig. 4.6** Circuito de acondicionamiento del luxómetro

Este sensor permite un rango de medida, véase la Fig. 4.7, desde 0 Lux, es decir oscuridad absoluta, hasta 100000 lux, luz solar en pleno mes de Agosto.



**Fig. 4.7** Relación entre intensidad e Iluminación

## CAPITULO 5. ALIMENTACIÓN DE LA ESTACIÓN METEOROLÓGICA

### 5.1 Alimentación de la placa

A la hora de plantear la realización del proyecto se ha tenido en cuenta que los componentes debían de tener un consumo lo más bajo posible. Para ello se tomaron diversas consideraciones.

El PIC debe ser de bajo consumo, como consecuencia se optó por un 16F877A-LF (su consumo es de 3.8 mA) que permite una relación aceptable de prestaciones y consumo. Mediante el PIC se activa:

- La alimentación del circuito de captación de los sensores, ya que de este modo se consigue desconectarlos cuando no estén en uso.
- La habilitación de la comunicación con el MODEM, ya que el chip de comunicaciones Max222 permite su habilitación o desconexión mediante una de sus entradas
- El control de la habilitación del MODEM.

La regulación de tensión a 5V se realiza mediante un regulador de bajo consumo MC78LC50NTRG, que tiene una corriente de fugas de tan solo 1.1 uA y por tanto, es ideal para alimentación de sistemas autónomos donde el ahorro de energía es fundamental.

El sistema tiene dos estados de funcionamiento; estado de reposo y el estado activo. En el estado de reposo únicamente funciona el microcontrolador reduciendo su consumo hasta unos 4 mA. Por otra parte, el sistema entra en estado activo durante unos dos minutos una vez que haya transcurrido un tiempo en estado de reposo. Inicialmente este tiempo es de 20 minutos pero puede aumentar si se detecta que la batería ha reducido su carga. El estado de carga se determina a partir de la tensión de la batería para un consumo aproximado de unos 100 mA. Nótese en la figura 5.1 como el microcontrolador puede inducir la descarga momentánea de la batería a través del MOSFET para así determinar el estado de carga.

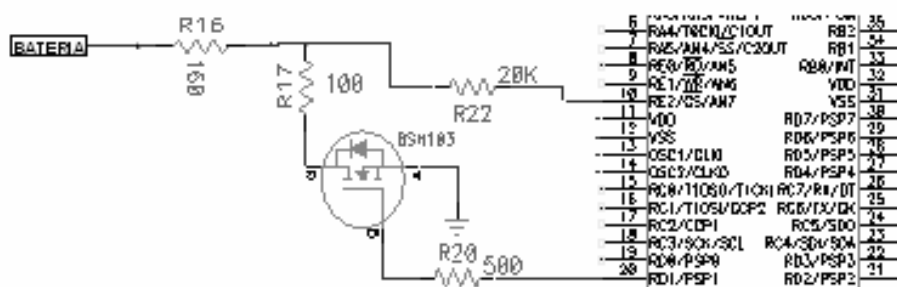


Fig. 5.1 Medición del estado de la batería

Durante el estado activo se conecta el modem, así como los sensores y sus circuitos acondicionadores. El consumo del sistema sube hasta unos 120 mA, siendo 110mA debidos al consumo del modem.

## 5.2 Elección de la batería

En primer lugar se estima el consumo diario de la placa, para ello se tiene en cuenta su funcionamiento durante 1 hora:

$$IAh = 54 \text{ min} * 4mA + 6 \text{ min} * 120mA = 15.6mAh \quad (5.2)$$

Si la batería a plena carga tiene una tensión de 12 V, la energía que necesitaría para el funcionamiento del sistema en un día será:

$$\begin{aligned} E_T &= IAh * h * V \\ E_T &= 15.6mAh * 24h * 12V \\ E &= 4.49Wh / \text{día} \end{aligned} \quad (5.3)$$

Una vez realizado el cálculo del consumo teórico, se calcula el consumo energético real, para hacer frente a los posibles factores de perdidas.

$$E = \frac{E_T}{R} \quad (5.4)$$

Donde R es el parámetro de rendimiento global de la instalación, definido como:

$$R = (1 - K_b - K_c - K_v) * \left( 1 - \frac{K_a * N}{P_d} \right) \quad (5.5)$$

Los factores de la ecuación 5.5 son los siguientes:

- Kb: Coeficiente de perdidas por rendimiento del acumulador  
0.05 en sistemas que no demanden descargas intensas  
0.1 en sistemas con descargas profundas
- Kc: Coeficiente de pérdidas en el convertidor:  
0.05 para convertidores senoidales puros, trabajando en régimen optimo  
0.1 en otras condiciones de trabajo
- Kv: Coeficiente de perdidas varias (rendimiento de la red):  
0.05-0.15 Agrupa otras perdidas
- Ka: Coeficiente de autodescarga diario:  
0.002 para baterías de baja autodescarga Ni-Cd  
0.005 para baterías estacionarias de Pb-ácido  
0.012 para baterías de alta autodescarga
- N: Número de días de autonomía de la instalación:  
Serán los días que la instalación deberá funcionar bajo una irradiación mínima o nula. 4 – 15 días.

- Pd: Profundidad de descarga diaria de la batería  
Esta profundidad de descarga no debe ser superior al 80%

Una vez escogidos los valores adecuados para el sistema, se deduce el siguiente rendimiento:

$$R = (1 - 0.05 - 0 - 0.05) * \left(1 - \frac{0.005 * 4}{0.7}\right) = 0.825 \quad (5.6)$$

Quedando la potencia real consumida:

$$E = \frac{4.49Wh / día}{0.825} = 5.44Wh / día \quad (5.7)$$

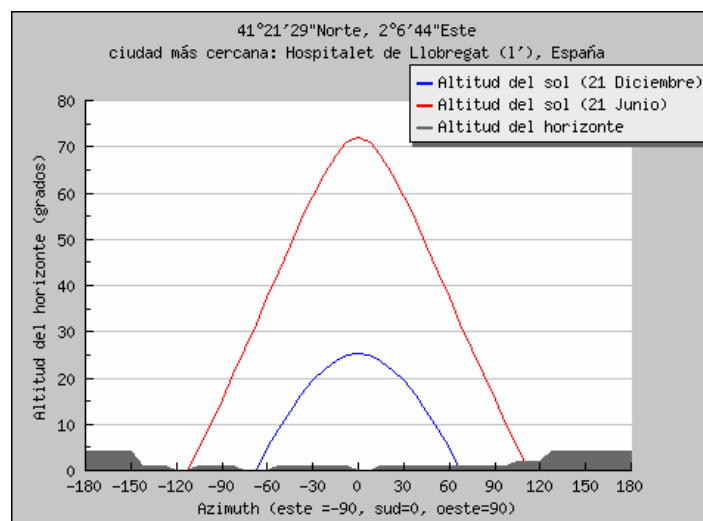
A partir de este valor se calcula la capacidad de la batería necesaria para su funcionamiento durante tres días con unas condiciones climáticas que incapacitarían la extracción de energía del panel.

$$C = \frac{E * N}{V * P_d} = \frac{3.49Wh * 3}{12V * 0.7} = 1.94Ah \quad (5.8)$$

La batería será el modelo np2-12 fabricante Yuasa, con una carga de 12V 2Ah

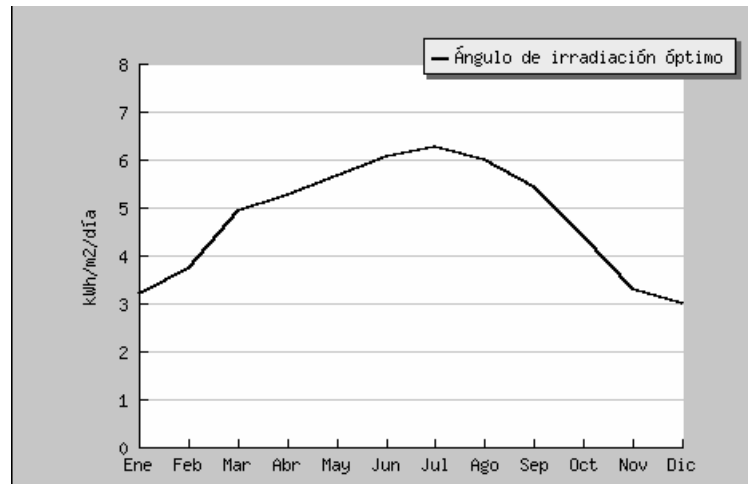
### 5.3 Elección del panel solar

Una vez seleccionada la batería, se ha de establecer cual es el tamaño de la placa solar. Para ello, se ha de tener en cuenta una serie de datos respecto a la ubicación del sistema. Las pruebas del sistema son realizadas en la localidad de Hospitalet de Llobregat, Barcelona. Se buscan los datos de mejor orientación de la placa solar en las peores condiciones posibles en dicho emplazamiento, véase la **Fig. 5.9**. Será en invierno, donde es necesario un mejor aprovechamiento de la cantidad de luz disponible.



**Fig. 5.9** Distribución de la altitud solar

En el caso que nos atañe, la inclinación óptima es de  $35^\circ$  y la orientación acimutal de  $-2^\circ$ . Utilizando esta orientación se obtiene unas horas de pico solar media anual de 4.7 horas. Es decir, la irradiación diaria media anual es equivalente a 4.7 horas a  $1000 \text{ W/m}^2$ .



**Fig. 5.10** Irradiación anual para el emplazamiento elegido

La placa solar de la que se dispone es el modelo C-0156 de Cebek Components. Tiene una tensión en circuito abierto de 12 V y suministra una potencia de 1.92W a plena iluminación, es decir  $100 \text{ W/m}^2$ . Considerando un 10% de pérdidas en la extracción de energía, se deduce la potencia media anual obtenida:

$$\begin{aligned}
 E_g &= 0.9 * W * HPS \\
 E &= 0.9 * 1.92W * 4.7 & (5.11) \\
 E &= 8.12W / \text{día}
 \end{aligned}$$

Nótese que la energía generada diaria es superior a la energía consumida por el sistema.

Para la protección del sistema, la placa solar tiene integrado un diodo, mediante el cual se evita la descarga de la batería. Asimismo se ha colocado un diodo zener de 13,5V en paralelo a la batería para evitar algún posible desperfecto por exceso de carga.

Las pruebas se han realizado durante la segunda semana del mes de agosto 2008, dejando el sistema en funcionamiento y orientado correctamente durante una semana de tiempo. Durante dicho intervalo no se apreció descarga del nivel de la batería. Habría que tener en cuenta que no son las peores condiciones de funcionamiento, siendo estos los meses de diciembre y enero. No obstante, el valor medio de HPS es de tres horas durante estos meses y por tanto, estarían también dentro de los márgenes para el correcto funcionamiento del sistema.

Se ha comprobado experimentalmente que la batería podría alimentar al sistema durante cinco días en condiciones meteorológicas adversas, es decir

cuando no hay suficiente iluminación para funcionar el panel solar. Nótese que este tiempo es superior a los tres días calculados teóricamente. Esto se debe a que en el calculo teórico se ha supuesto que el estado activo se producía cada veinte minutos. Sin embargo, a medida que se descarga la batería este tiempo se alarga y provoca una mayor duración de la batería.



## CAPITULO 6. SISTEMA DE TRANSMISIÓN DE DATOS

Se requiere que la estación meteorológica sea fácilmente accesible por los usuarios y estos puedan ser identificados. Al ser un sistema autónomo es necesario que dicho sistema tenga gran cobertura en el territorio español. Al no requerirse la transmisión de gran cantidad de datos ni grandes anchos de banda, debe ser lo más simple posible y económicamente viable. Considerando esta requisitos se optado por la transmisión mediante un modem GSM.

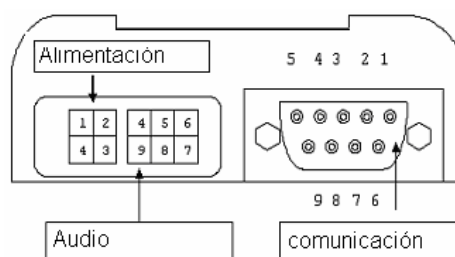
### 6.1 Características del MODEM GSM

Este modelo es el más sencillo y económico de la casa Audiotel. No obstante, ofrece una amplia gama de prestaciones, como comunicaciones de voz, fax y mensajería SMS, que serán suficientes para la realización del proyecto, el cual se centra en el envío de sms.

Las **Fig. 6.1** y **Fig. 6.2** muestran el aspecto físico del módem, así como un detalle de sus puertos externos.



**Fig. 6.1** Aspecto exterior MODEM



**Fig. 6.2** Puertos MODEM

Existe tres tipos de puertos; puerto de alimentación, puerto de audio, puerto para la transmisión y recepción de datos ( RSR232 ).

El puerto de alimentación consta de 4 pines, de los cuales, sólo son útiles tres; la masa ( GND ), tensión de alimentación ( Power IN ) y el encendido ( ING ). Véase tabla 6.1.

**Tabla 6.1**

CONECTOR	PIN	SEÑAL	USO	COLOR
4 PIN MICROFIT	1	Power IN	Power IN (+)	RojoOUT
	2	GND	Tierra	Negro
	3	IGN	Encendido	Amarillo
	4	PDN	Power Down	N.E

El módem permite tensiones de alimentación comprendidas entre 8 a 32V.

El puerto de comunicación del módem GSM se compone de un conector de 9 pines. Este está constituido para poder ser conectado a cualquier tipo de

periférico, como puede ser un PC o cualquier otro tipo de controlador. En el proyecto solamente se han utilizado los pines TX y RX. Por este motivo, se ha cortocircuitado los pines 1,4,6 del puerto serie del MODEM. La tabla 6.2 muestra la función de cada uno de los terminales.

CONECTOR	PIN	EIA	USO	IN/OUT
<b>DB9 SERIAL RS232</b>	<b>1</b>	<b>DCD</b>	<b>DataCarrier</b>	<b>OUT</b>
	<b>2</b>	<b>RX</b>	<b>Recep Data</b>	<b>OUT</b>
	<b>3</b>	<b>TX</b>	<b>Trans Data</b>	<b>IN</b>
	<b>4</b>	<b>DTR</b>	<b>DataTerm.Ready</b>	<b>IN</b>
	<b>5</b>	<b>GND</b>	<b>Ground</b>	
	<b>6</b>	<b>DSR</b>	<b>DataSetReady</b>	<b>OUT</b>
	<b>7</b>	<b>RTS</b>	<b>RequesttoSend</b>	<b>IN</b>
	<b>8</b>	<b>CTS</b>	<b>Clear to Send</b>	<b>OUT</b>
	<b>9</b>	<b>RI</b>	<b>Ring ind.</b>	<b>OUT</b>

**Tabla 6.2** Función de los terminales

Para utilizar el módem en una aplicación concreta deben configurarse algunas de sus características. Concretamente, deben sincronizarse las velocidades del puerto RS232 con la del controlador y el modo de configuración de los mensajes SMS.

Existen dos modos de configuración; modo PDU y modo TXT.

Por cuestiones de simplicidad en la codificación se ha optado por la opción en modo texto. En la **Fig. 6.3** se muestra como sería la respuesta del módem GSM para un mismo mensaje de prueba.

<p>Respuesta del módem en modo TXT:</p> <p>+CMGR: "REC READ", "+34670341212", "05/05/10,16:54:28+04"</p> <p><b>Mensaje de prueba</b></p> <p>Respuesta del módem en modo PDU:</p> <p>+CMGR: 1,,34</p> <p><b>07914306073011F0040B914376301412F200005050016145824011C</b></p> <p><b>DB27B1E</b></p> <p><b>569741E432082EAF97C561</b></p>
---

**Fig. 6.3:** Diferencia entre la recepción de mensajes en modo TXT y PDU

## 6.2 Gestión del MODEM GSM

El control del GSM BASE por parte del microcontrolador se realiza a partir del puerto RS232 mediante los denominados comandos AT. Estos comandos se identifican con una cadena de caracteres en código ASCII.

En la tabla 6.3 se muestra una breve selección de los comandos utilizados.

**Tabla 6.3**

Codificación en ASCII	Descripción	Ejemplo
AT+CMGF= <i>X</i>	Configuración en modo PDU (si <i>X</i> =0) o en modo TXT (si <i>X</i> =1)	AT+CMGF=1
AT+CMGS	Mandar mensaje de texto	AT+CMGS=+34"teléfono" <b>ENTER</b> > Mensaje <b>ctrl.-Z</b>
AT+CMGR= <i>X</i>	Leer el mensaje recibido Numero <i>X</i>	AT+CMGR=1
AT+CMGD= <i>X</i>	Borrar mensaje numero <i>X</i>	AT+CMGD=1

## 6.3 Conexión con la estación meteorológica

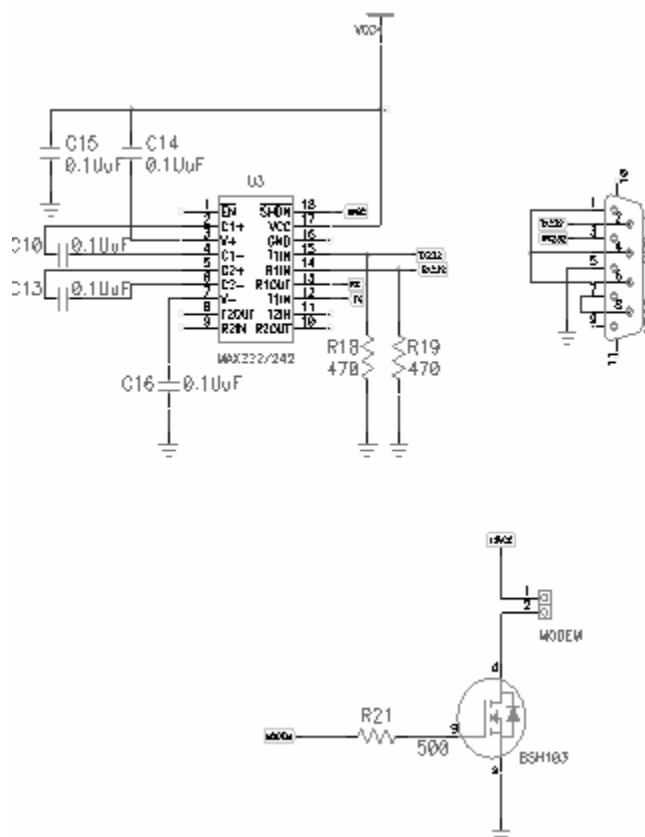
El microcontrolador 16F877A i/p dispone de terminales para la transmisión de datos serie que se ajusta al protocolo de comunicaciones RSR232. El único problema para la interconexión entre estos terminales y los del puerto serie del módem reside en la diferencia entre sus niveles lógicos. Para solventar este inconveniente se ha utilizado el MAX222 de Maxim. Se ha elegido debido a que puede desconectarse mediante uno de sus puertos para así no consumir mientras no este en uso por el sistema.

**Fig. 6.4** Esquema de conexionado RS232 al PIC

## CAPITULO 7. DISEÑO DE LA PLACA

### 7.1 Esquema general

Para la realización de la placa se ha elegido el programa P-cad 2002 que integra el esquemático. En la **Fig. 7.1** podemos observar la conexión del chip de comunicaciones así como la alimentación del MODEM. En la **Fig. 7.2** es posible ver la comunicación con el programador vía rj-45, así como los diferentes sensores, su acondicionamiento de señal, filtros, alimentación del sistema y control de la alimentación de diferentes partes del circuito.



**Fig. 7.1** Esquemático transmisión rs232 y control alimentación MODEM

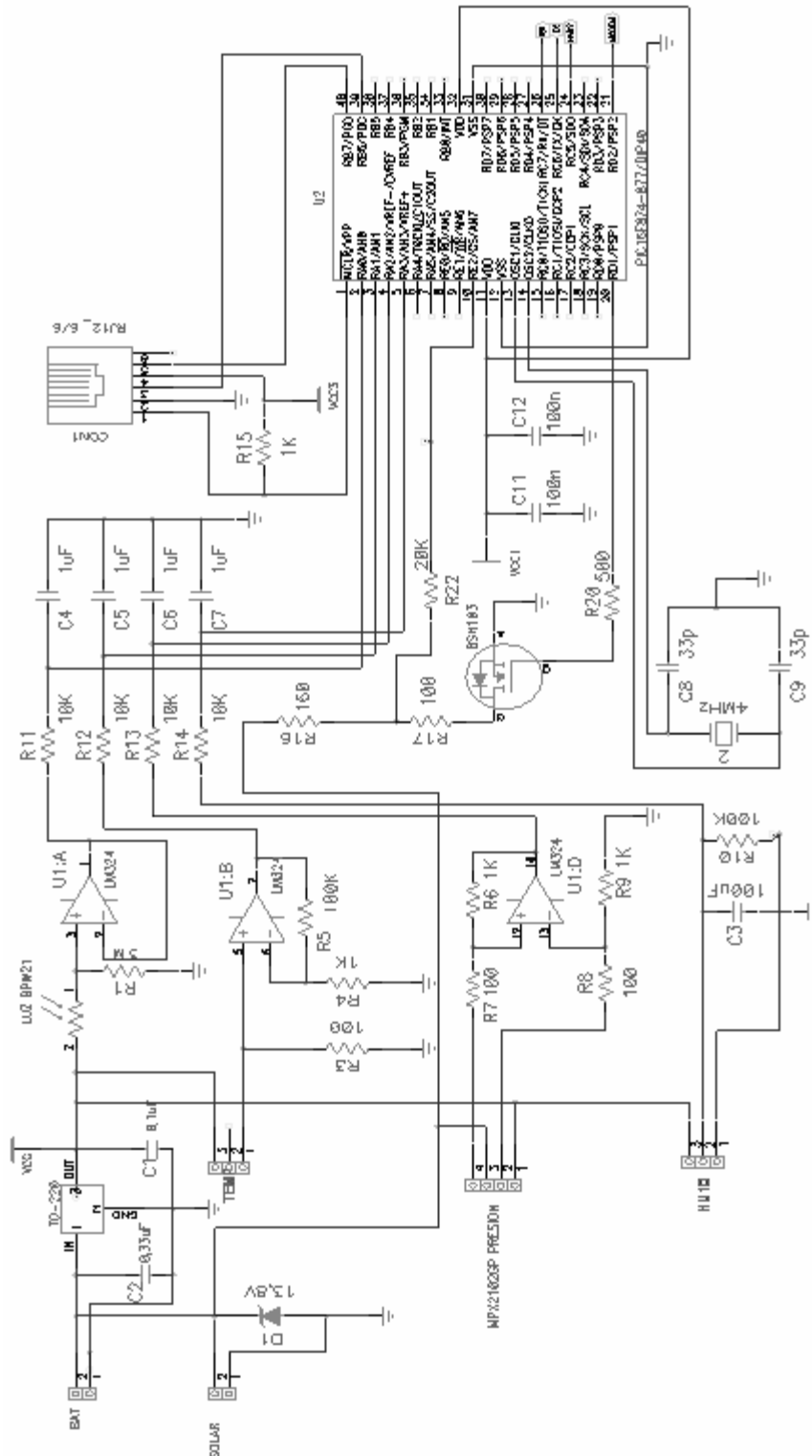
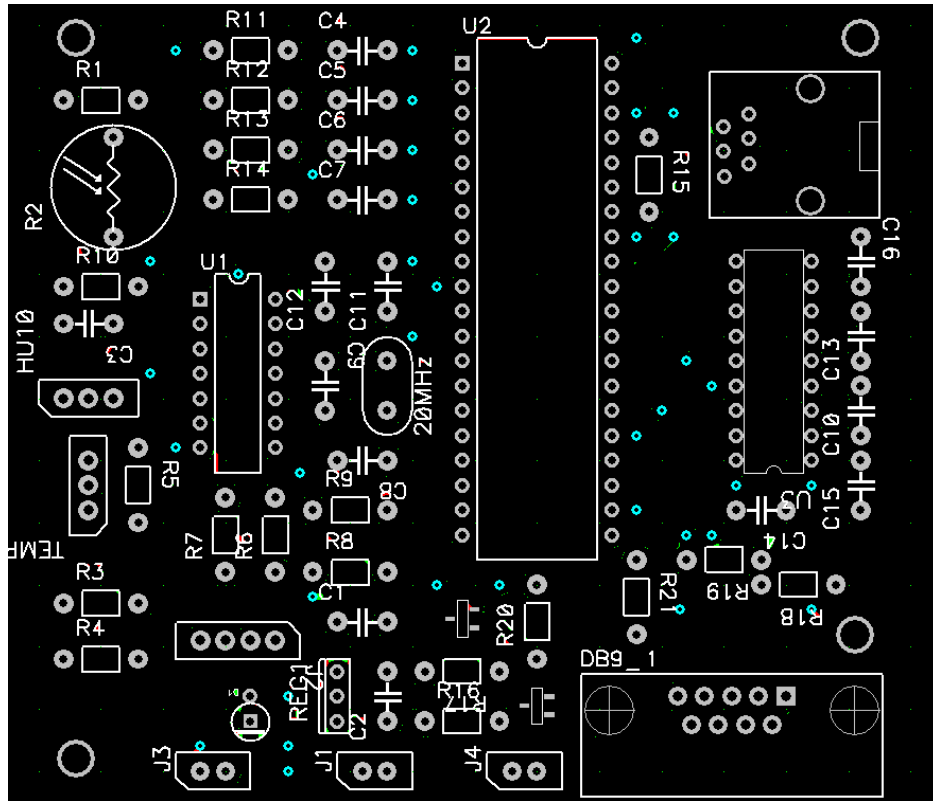


Fig. 7.2 Esquema sensores, alimentación y PIC

## 7.2 Diseño del circuito impreso.

Una vez definido el esquemático se importa al programa del P-CAD para el diseño del pcb, en primer lugar se escoge la posición de los componentes para intentar minimizar el tamaño de la placa **Fig. 7.3**.



**Fig 7.3** Distribución de los componentes en la placa

Para realizar las conexiones, se importa el esquemático. El trabajo se simplifica debido a que mediante el programa PCAD se realiza la conexión de los componentes y únicamente se han de modificar el lugar por donde pasan las pistas quedando el diseño como se muestra en las figuras **Fig. 7.4** y **Fig. 7.5**

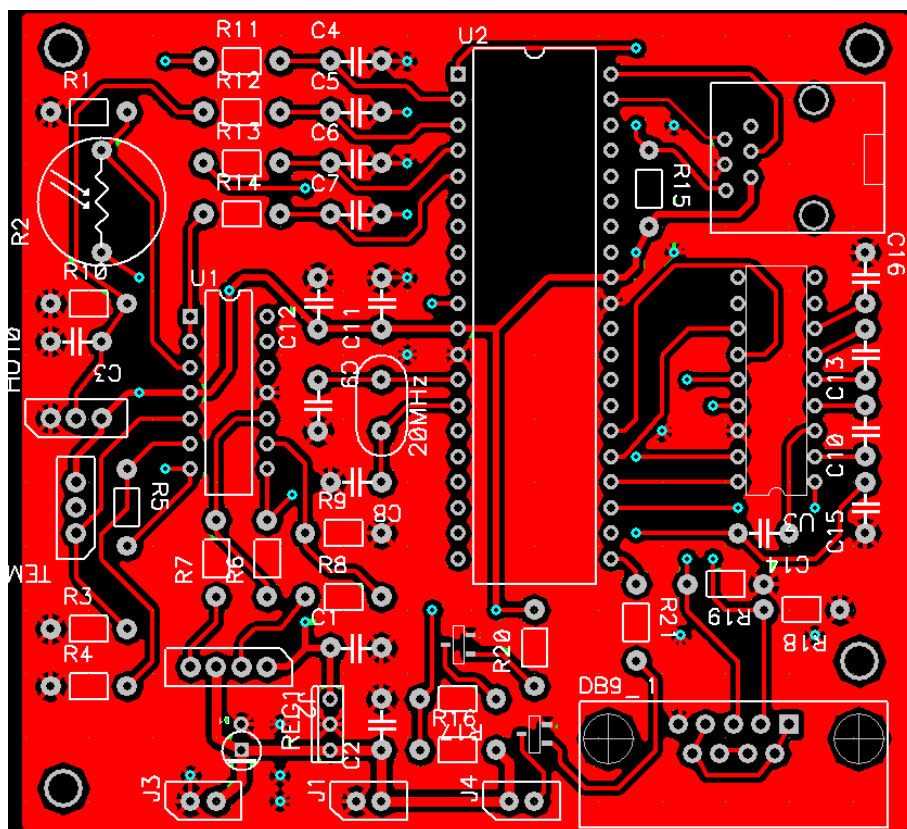


Fig. 7.4 Cara anterior PCB

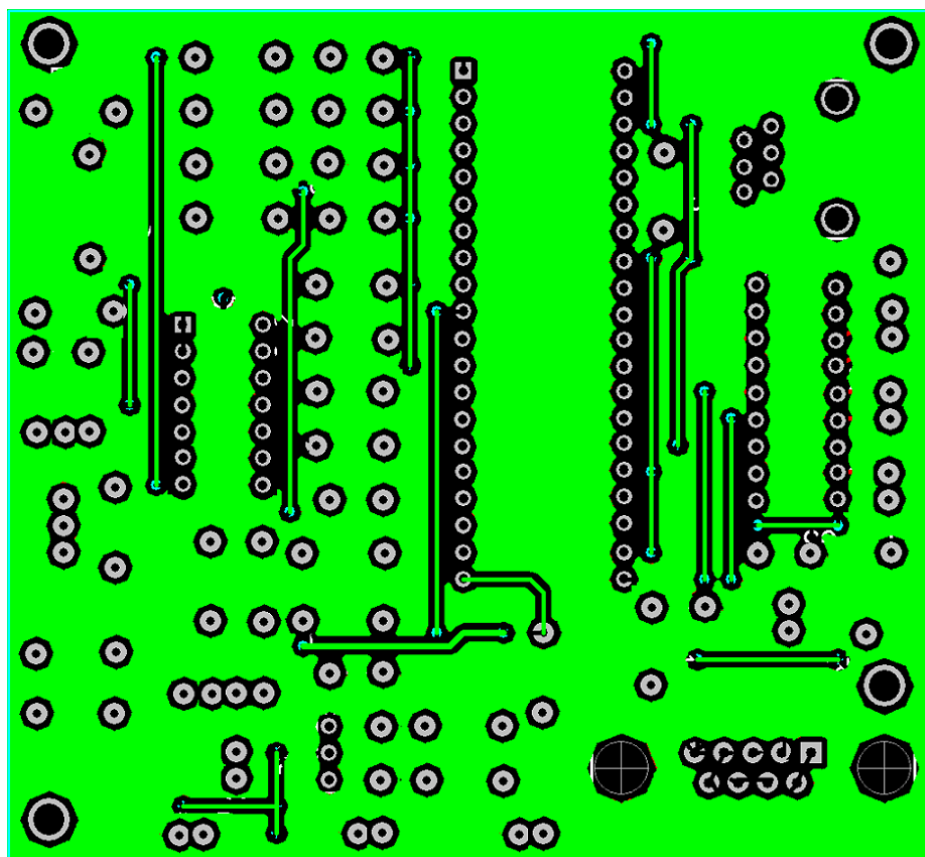


Fig. 7.5 Cara posterior PCB



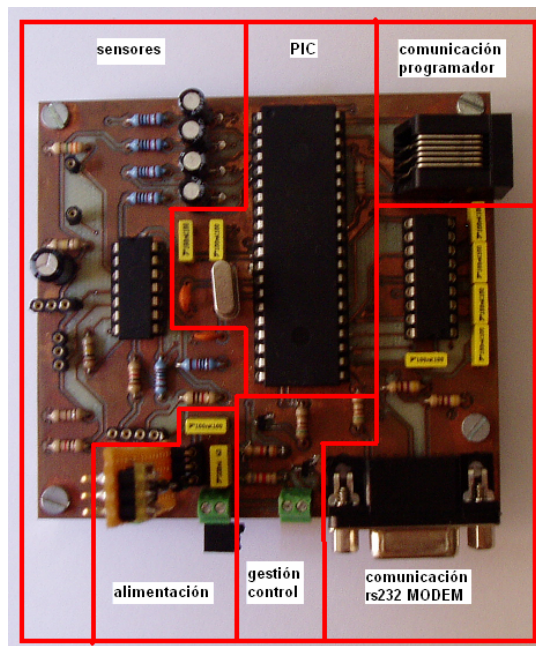
### 7.3 Aspecto terminado del circuito

Una vez realizada la placa y colocados los componentes el resultado final es el mostrado en la **Fig. 7.6**



**Fig. 7.6** Aspecto final del proyecto

En la placa se puede observar los diferentes bloques que forman el sistema. Véase la **Fig. 7.7**



**Fig. 7.7** Distribución de los componentes en la placa

## CONCLUSIONES

La finalidad del proyecto ha sido la creación y el desarrollo de una estación meteorológica autónoma y de fácil transporte, para su ubicación en cualquier lugar sin dependencia de ningún factor ya sea de comunicaciones o de alimentación.

El enfoque inicial del proyecto se centró principalmente en la autonomía del sistema, línea que se ha seguido durante el transcurso de su desarrollo. Los componentes fueron seleccionados con el objetivo de minimizar su consumo. Aun así, gran parte del diseño ha sido condicionado por el elevado consumo del modem, habiendo de ajustar el tiempo de conexión de componentes a fin de poder compensarlo y pretendiendo que la placa solar no fuera de unas dimensiones incómodas para su transporte.

Mediante el proyecto se han podido observar diferentes apartados de análisis, diseño, elaboración de placas y mediciones.

El diseño se podría ampliar en un futuro mediante memorias externas que permitieran la incorporación de un mayor número de usuarios. Otra posible mejora sería un control más eficaz de los usuarios a través de un administrador. La posibilidad de Introducir más variables meteorológicas o en su defecto, establecer una red de estaciones meteorológicas que se comunicaran entre ellas y que el sistema fuera capaz de hacer predicciones simples en función de los parámetros captados, serían posibles mejoras o ampliaciones.

Dichas posibles ampliaciones afectarían al consumo del circuito, debido a que los periodos de conexión serían más largos, pudiéndose optimizar la energía captada por la placa solar mediante su orientación automática para una óptima orientación en cada momento.

## BIBLIOGRAFÍA

- [1] <http://www.microchip.com>
- [2] <http://studies.ac.upc.edu/EPSC/SED/>
- [3] <http://todopic.mforos.com/?cat=327473>
- [4] <http://sunbird.jrc.it/pvgis/apps/pvest.php?lang=en&map=europe>
- [5] <http://www.unav.es/cti/curso-c/>
- [6] <http://www.alldatasheet.com>



Escola Politècnica Superior  
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANEXOS

**Título:** Diseño de una estación meteorológica autónoma gestionada mediante mensajes SMS

**Titulación:** Ingeniería Técnica de Telecomunicaciones, especialidad Sistemas de Telecomunicaciones

**Autor:** Jorge Luján de Francisco

**Director:** Oscar López Lapeña

**Fecha:** 4 de diciembre 2008

## 10 Funcionamiento IC2 y MPLAB

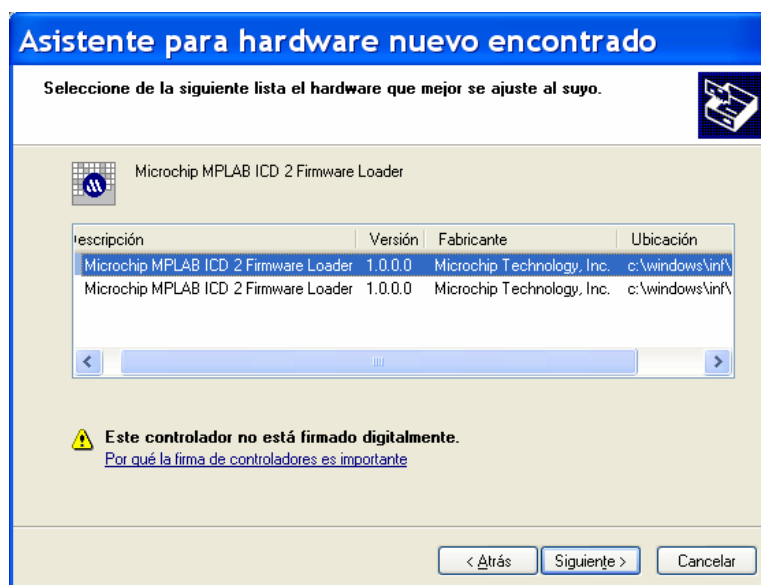
En el primer anexo, se centra en la instalación y configuración del programa MPLAB y e ICD2.

Procedemos a la instalación del programa MPLAB descargado de la web de microchip, ejecutando el asistente, véase la **Fig. 10.1**.



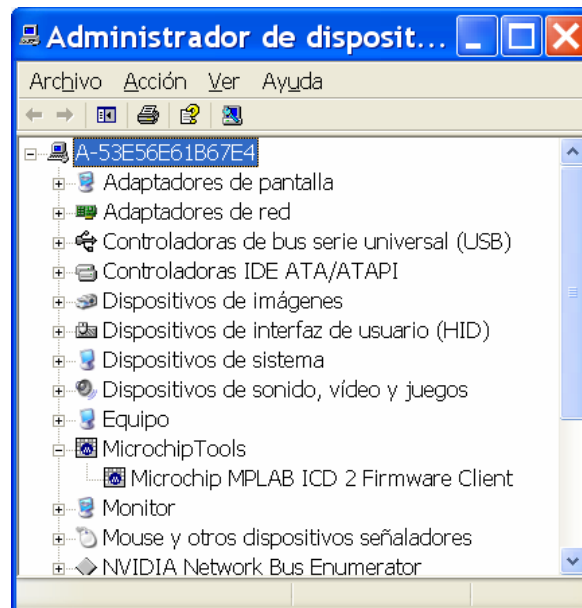
**Fig. 10.1** Instalación MPLAB

Al realizar la conexión vía usb del icd2, el asistente de nuevo hardware encontrado nos informa del nuevo dispositivo. Se habrá de instalar los 2 controladores que acompañan al programador. Véase la **Fig. 10.2**, dichos controladores los podemos encontrar dentro de la carpeta de instalación del MPLAB



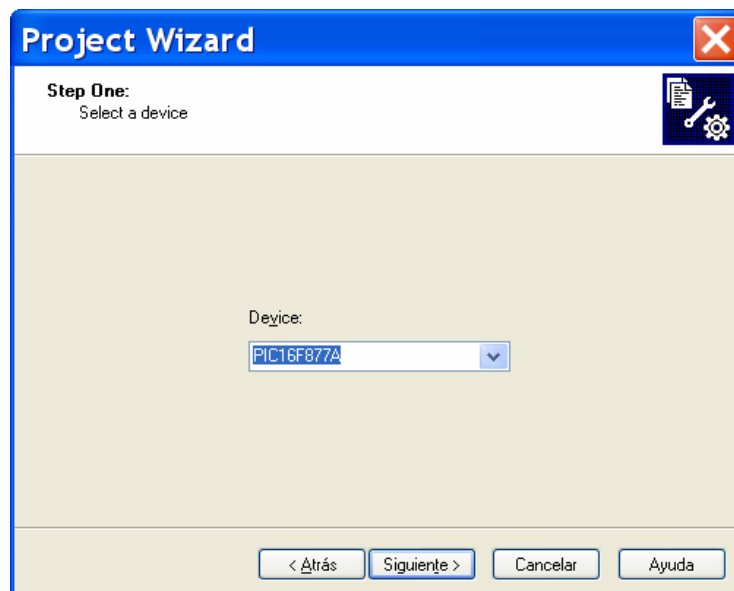
**Fig. 10.2** Instalación de los controladores ICD2

Una vez debidamente instalado el dispositivo, en el administrador de dispositivos comprobaremos su correcto funcionamiento, véase la **Fig. 10.3**



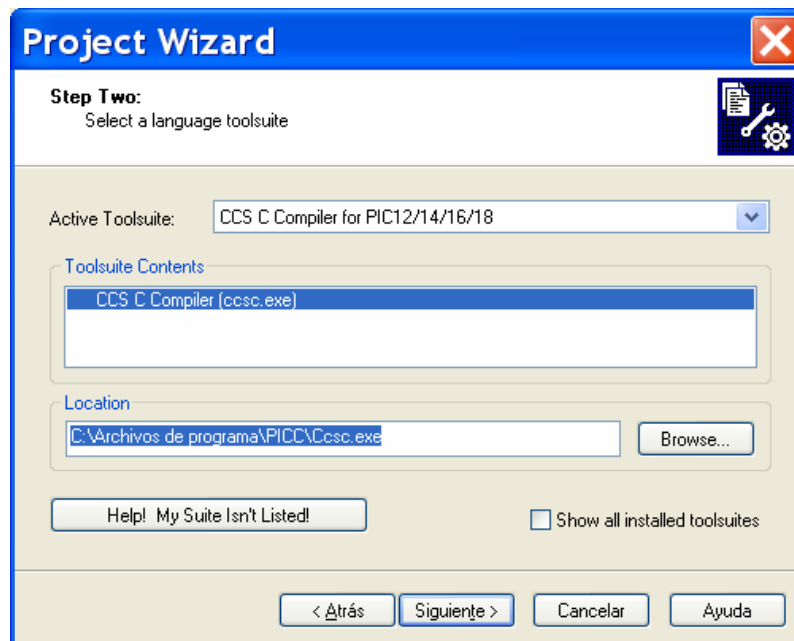
**Fig. 10.3** Comprobación de la instalación de controladores

Ejecutamos el programa MPLAB y creamos un nuevo proyecto, definiendo el PIC que utilizaremos, en este caso PIC16f877A, véase la **Fig. 10.4**



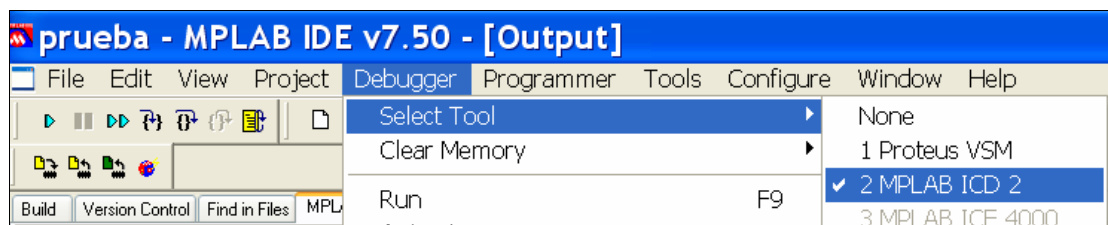
**Fig. 10.4** Selección del PIC

Otro paso importante es la elección del lenguaje de programación, en este caso, se ha elegido CSS, véase **Fig. 10.5**.



**Fig. 10.5** Elección del lenguaje de programación.

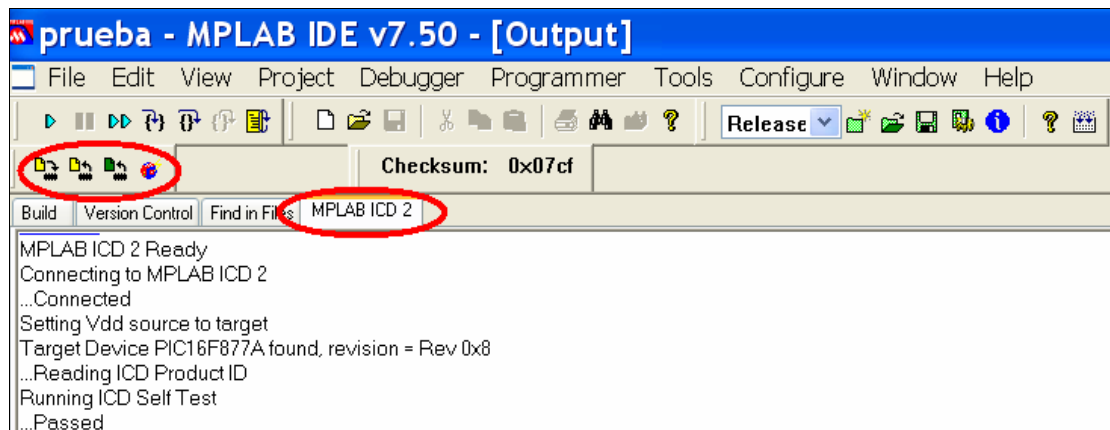
Ya creado el proyecto, se selecciona el dispositivo para realizar la compilación y pruebas mediante *Debugger>select tool>icd2* seleccionamos el nuestro véase **Fig. 10.6**



**Fig. 10.6** Elección del ICD2

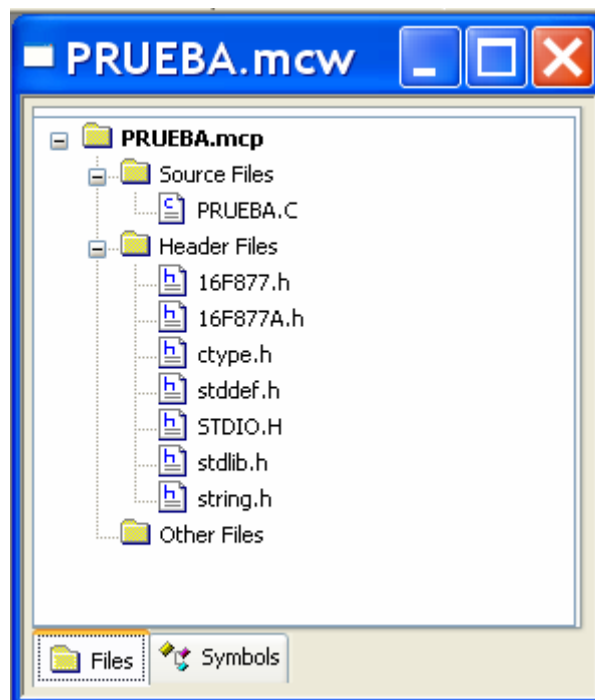
Las opciones del *debugger* nos permiten insertar puntos de parada del programa, ejecutarlo paso a paso, continuo, leer el programa del PIC, también habrá opciones interesantes como *Watch*, en la cual podemos ver las variables empleadas y sus cambios de estado a tiempo real.

Una vez instalado el ICD2 observamos que aparecen nuevas opciones en el MPLAB para la gestión del programador, véase **Fig. 10.7**



**Fig. 10.7** Nuevas opciones en el MPLAB

El proyecto consta de un programa principal, y librerías para la interpretación de las instrucciones del programa principal, también llamadas cabeceras. Véase la **Fig. 10.8**



**Fig. 10.8** Programa y librerías

Dichas librerías tienen una función específica dentro del programa

**16f877.h** contiene la definición de los pines de conexión del PIC

**String.h** contiene la definición de macros, constantes, funciones y tipos de utilidad para trabajar con cadenas de caracteres y algunas operaciones de manipulación de memoria

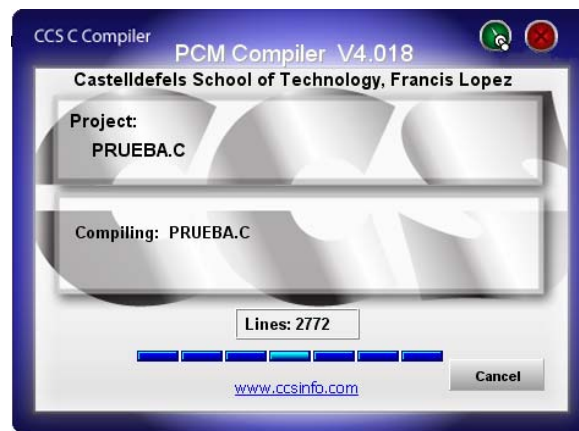


**Ctype.h** fue diseñado para operaciones básicas con caracteres. Contiene los prototipos de las funciones y macros para clasificar caracteres.

**Stdlib.h** contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.

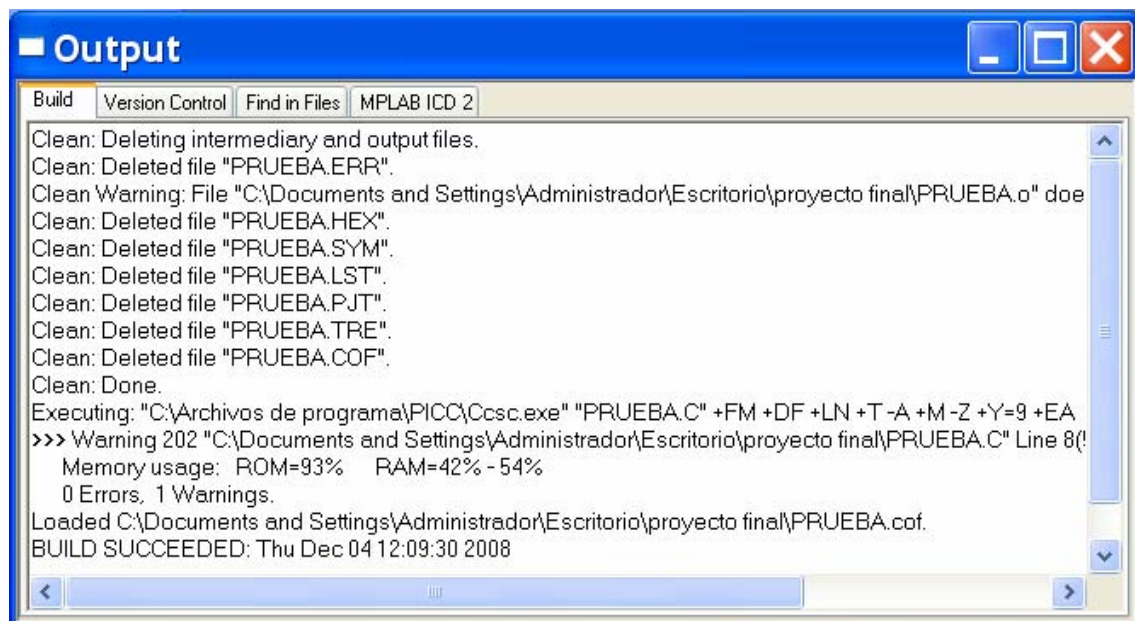
**Stdio.h** contiene las definiciones de macros, las constantes, las declaraciones de funciones y la definición de tipos usados por varias operaciones estándar de entrada y salida.

A la hora de compilar el proyecto se realiza mediante el compilador seleccionado con anterioridad, dicho proceso es automático al realizar una compilación. Véase la **Fig. 10.9**



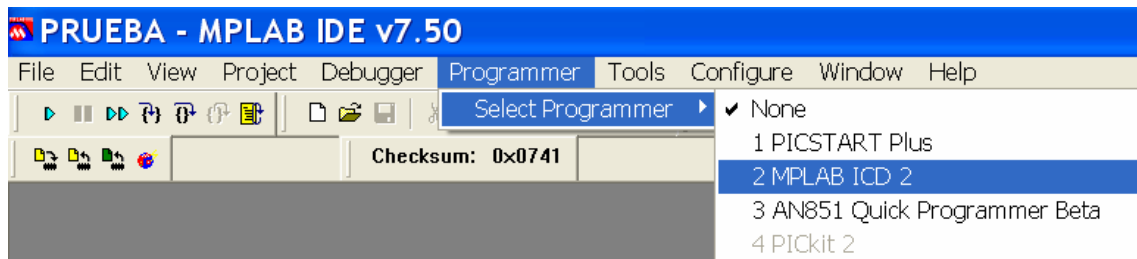
**Fig. 10.9** Compilación mediante css

Una vez realizada la compilación el sistema nos informa de posibles errores, alertas y la cantidad de memoria del sistema ocupada. Véase **Fig. 10.10**



**Fig. 10.10** Proyecto compilado

Una vez realizada la compilación del proyecto, configuraremos el programador para la grabación en el PIC, véase la **Fig 10.11**, se procede a la desconexión del programador, el PIC ya estaría grabado y listo para su funcionamiento.



**Fig. 10.12** Grabación en el PIC

## 11 Programación estación meteorológica en C

```
#include <16F877A.h>
#define ADC=8
#include <STDIO.H>
#include <string.h>
#include <stdlib.h>
#define XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#define delay(clock=4000000) // Oscilador a 4 Mhz
#define rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, brghlok,ERRORS) // RS232 Estándar
#define standard_io(b)

// VARIABLES EN RAM ///////////////////////////////////////////////////////////////////

int1 flagcommand=0;
int usuario_valido=0;
short int borrar=0;
int i,j,k,f,g,s,r;
int telf;
int sms_nulo=0x00;
int sms_devuelto=0;
int cod;
int A=1;
int B=0;
long int C=0;
int habilita=1;
long int D=0;
int X=40;
long int h;
float valor;
int adc_humedad=0x00;
int adc_temperatura=0x00;
long int adc_luz=0x00;
float adc_presion=0x00;
float adc_bateria=0x00;
char num_modem;
char rcvchar=0x00;
char num[11];
char sms[38];
char gestion[12];
char inicio[3]="34";
char caracter;

// CONSTANTES ///////////////////////////////////////////////////////////////////

const char eliminar[5]="Baja";
const char autoriza[5]="Alta";
const char temperatura[12]="Temperatura";
const char presion[8]="Presion";
const char humedad[8]="Humedad";
const char luz[4]="Luz";
const char borrado[12]="000000000000";
const char alarmas[7]="Alarma";
const char alarma_temp[12]="Alarma.Temp";
const char alarma_luz[12]="Alarma.Luz";
const char alarma_pres[12]="Alarma.Pres";
const char alarma_hume[12]="Alarma.Hume";
```

```
//EEPROM TELF Localizaciones////////////////////////////////////

#define ROM_TELF1      0
#define ROM_TELF1_SIZE 11

#define ROM_TELF2      16
#define ROM_TELF2_SIZE 11

#define MAX222 PIN_C5
#define MODEM PIN_D2
#define SENSORES PIN_D0
#define BAT PIN_D1

// Declaración de Estructuras //////////////////////////////////

struct panel_alarma
{
    long int luz_max;
    long int luz_min;
    int temp_max;
    int temp_min;
    int presion_max;
    int presion_min;
    int humedad_max;
    int humedad_min;
};

struct telf_alarmas
{
    struct panel_alarma telf[3];
};

struct telf_alarmas alarma;

// Declaración de Funciones //////////////////////////////////

void inicbuff(void);
void arranque_modem(void);
void addcbuff(char c);
void procesa_sms(void);
void anadir_telf(void);
void comprobar_borrar_telf(void);
void eliminar_sms(void);
void toma_adc(void);
void busca_telf(void);
void sms_toma(void);
void compara_alarma_adc(int m);
void envio_sms(void);
void alarmas_valor_max_min(int v);
void tipo_alarma(void);
```

```

// INTERRUPTACIONES //////////////////////////////////////

#int_rda
void serial_int() { // Interrupción recepción serie USART
    rcvchar=0x00; // Inicializo caracter recibido
    if(kbhit()){ // Si hay algo pendiente de recibir ...
        rcvchar=getc(); // lo descargo y ...
        character=rcvchar; // lo añado al buffer y ...
        if(character=='+') //detecta sms
            {g++;}
        if(g==1)
            {s++;}
        if(s==8)
            { if(character=='0')
                { disable_interrupts(int_rda); //deshabilita
                  disable_interrupts(global); //interrupción rda
                  break;
                }
            }
    }
    if(k<37&&0<g)
    {switch(character){
        case '"':
            i++; // Si ha recibido un " aumenta el contador
        default:
            if(i==3)
                {num[(j++)-2]=character;} // Añade caracter a num
            if(i==6)
                {sms[(k++)-3]=character; // Añade caracter a sms
            if (character=='K') // final de mensaje
                {sms_nulo=1; // Si, hay sms
                  break;}
            }
        }
    }
    if(character=='K'&&sms_nulo==1)
    { gcommand=1; //activa procesa_sms
      disable_interrupts(int_rda); //deshabilita
      disable_interrupts(global); //interrupción RDA
      break;
    }
}

#int_RTCC // Interrupción desbordamiento timer
void RTCC_isr() { // del TIMERO RTCC
    if( D>909){ // (909*0.033*X)/60= X min
        if(C > X)
        { habilita=0; //habilita enciendo modem
          D=0; // inicia las variables contador
          C=0;
          disable_interrupts(int_RTCC); // deshabilita
        } // interrupción
        ++C; // timer
        D=0;
    }
    ++D;
}

```

```

// Programa Principal ////////////////////////////////////////

void main() {

    inicbuff(); // inicializar las variables
    num_modem=0; // Borra buffer al inicio
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(ALL_ANALOG); // config puertos analog
    setup_counters(RTCC_INTERNAL,RTCC_DIV_128); // TIMER0: Clock Interno,
    // Prescaler 128
    setup_timer_1(T1_DISABLED); // para 1 RTCC cada 33.3 miliseg
    setup_timer_2(T2_DISABLED,0,1); // -> 1 Segundo = 30 RTCC
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
    enable_interrupts(int_RTCC); // Habilitar Interrup timer
    enable_interrupts(global);
    output_low (MAX222); // Deshabilitación
    output_low (MODEM); // alimentaciones
    output_low (SENSORES);
    output_low (BAT);
    X=40; // X= 20 min
    A=1; // inicialización lectura sms
    delay_ms(5000);
    inicio:
        if(habilita==0)
        { arranque_modem();
          if(flagcommand) procesa_sms();
        }
    goto inicio;
}

// DESARROLLO DE LAS FUNCIONES ////////////////////////////////////////

// Inicializar las variables y buffers//////////////////////////////////

void inicbuff(void) { // Inicia a \0 cbuff -----

    for(i=0;i<11;i++)
        { num[i]=0x00; } // a 0 el buffer de num
    for(i=0;i<38;i++)
        { sms[i]=0x00; } // aa 0 el buffer de sms

    i,j,k,f,s,g,r=0x00; // inicializa variables
    sms_nulo=0x00;
    sms_devuelto=0;
    flagcommand=0;
}

```

```

// Procesar SMS ////////////////////////////////////////

void procesa_sms(void) {

    flagcommand=0;           // Desactivo flag de comando pendiente.
    if(sms_nulo==0)
    {
        B++;                // cuenta sms vacios
        A++;                // contador de sms
        if(B==3)
        {
            habilita=1; //deshabilita encendido modem
            A=1;         // reinicia el contador de sms
            D,C,B=0;     // reinicia variables
            output_low (MODEM); // desconecta modem
            output_low (MAX222); // desconexion rs232
            enable_interrupts(int_RTCC); // habilitación
            enable_interrupts(global); // interrup timer
        }
        busca_telf();        // busca alarmas usuarios
        if(sms_devuelto>9)   // ¿hay alguna alarma activada?
        {
            envio_sms();     // envio de aviso al usuario
        }
    }
    if(sms_nulo==1)
    {
        eliminar_sms();      // elimina sms del modem
        A++;                 // aumenta contador de sms
        comprobar_borrar_telf(); // comprobación si el usuario
        strcpy(gestion , autoriza); // esta registrado
        if(!strcmp(sms, gestion,4)&& usuario_valido==0)
        {
            anadir_telf();    // nueva alta
        }
        if(usuario_valido==1)
        {
            setup_adc(ADC_CLOCK_INTERNAL);
            setup_adc_ports(ALL_ANALOG);
            toma_adc();        // captura de los parametros
            sms_toma();        // comprobación tipo sms
            strcpy(gestion , alarmas); // si el sms es una alarma
            if(!strcmp(sms, gestion,4))
            {
                tipo_alarma(); // introducción de la alarma
            }
            envio_sms();      // envio de sms
            busca_telf();     // busca alarmas usuarios
            if(sms_devuelto>9) // ¿alguna alarma activada?
            {
                envio_sms();  // envio de aviso al usuario
            }
        }
    }
    inicbuff();
}

// Eliminar SMS ////////////////////////////////////////

void eliminar_sms(void) {
    delay_ms(50);
    printf("\r\AT+CMGF=1\r\n");
    delay_ms(1000);
    printf("\r\AT+CMGD="); // eliminar mensaje del modem
    printf("%d",A);        // posición A
    printf("\r\n");
}

```

```
// Inicializar el modem ////////////////////////////////////////

void arranque_modem(void) {
    if(B==0)
    {
        output_high (MODEM);           // encendido modem
        output_high (MAX222);          // encendido rs232
        delay_ms (10000);
    }
    enable_interrupts(int_rda);         // Habilita Interrup RDA
    enable_interrupts(global);
    printf("\r\ATE0\r\n");
    printf("\r\AT+CMGF=1\r\n");         // Modem formato texto
    delay_ms(1000);
    printf("\r\AT+CMGR=");              // Leer mensaje modem
    printf("%d",A);                     // posición A
    printf("\r\n");
}

// Leer teléfono de la ROM ////////////////////////////////////////

void leer_eeprom_string(char * array, int8 address, int8 max_size)
{
    int8 i=0;

    *array=0;

    while (i<max_size)
    {
        *array=read_eeprom(address+i);
        if (*array == 0) {i=max_size;}
        else {
            array++;
            *array=0;
        }
        i++;
    }
}

// Escribir teléfono en la ROM ////////////////////////////////////////

void escribir_eeprom_string(char * array, int8 address, int8 max_size)
{
    int8 i=0;

    while (i<max_size) {
        write_eeprom(address+i,*array);
        if (*array == 0) {i=max_size;}
        array++;
        i++;
    }
}

```



```

// Comprobar borrar telf ////////////////////////////////////////

void comprobar_borrar_telf(void)
{
    telf=0; // inicialización de las variables
    usuario_valido=0;
    sms_devuelto=0;
    leer_eeprom_string(gestion, ROM_TELF1, ROM_TELF1_SIZE); //lee telf 1 ROM
    if(!strcmp(num, gestion,11)) // compara 1 telf ROM con el recibido
    {telf=1;}
    leer_eeprom_string(gestion, ROM_TELF2, ROM_TELF2_SIZE); //lee telf 2 ROM
    if(!strcmp(num, gestion,11)) // compara 2 telf ROM con el recibido
    {telf=2;}
    strcpy(gestion, eliminar); // comprueba si es petición baja usuario
    if(!strcmp(sms, gestion,4))
    { borrar=1;}
    if(borrar==1&& telf!=0)
    {
        strcpy(gestion, borrado);
        if(telf==1) // elimina el usuario 1 y sus alarmas
        {escribir_eeprom_string(gestion, ROM_TELF1, ROM_TELF1_SIZE);
        alarmas_valor_max_min(telf);
        }
        else if(telf==2) // elimina el usuario 1 y sus alarmas
        {escribir_eeprom_string(gestion, ROM_TELF2, ROM_TELF2_SIZE);
        alarmas_valor_max_min(telf);
        }
        sms_devuelto=1; // "usuario borrado"
        borrar=0;
        break;
    }
    else if (telf!=0)
    {
        usuario_valido=1;
        break;
    }
    else if (telf==0)
    {
        sms_devuelto=4; // "usuario no registrado"
    }
}

// coloca las alarmas de un usuario a los valores máximos ////////////////////////////////////////

void alarmas_valor_max_min(int v)
{
    alarma.telf[v].temp_max=318;
    alarma.telf[v].temp_min=253;
    alarma.telf[v].presion_max=2000;
    alarma.telf[v].presion_min=0;
    alarma.telf[v].luz_max=99999;
    alarma.telf[v].luz_min=0;
    alarma.telf[v].humedad_max=99;
    alarma.telf[v].humedad_min=0;
}

```

```
// Tomar mediciones sensores y estado de la batería //////////////////////////////////

void toma_adc(void) {

    output_high (SENSORES);           // habilitación sensores
    output_high (BAT);                // habilitación test de batería
    delay_ms(1);                      // habilitación acondicionamiento

    set_adc_channel(7);               // Lectura del canal 7 -> batería
    delay_ms(1);
    valor=read_adc();
    adc_bateria=valor/18.5;
    valor=0;
    delay_ms(1);

    if (adc_bateria<9)                // valor de X para la temporización
        {X=240;}                     // en función de la batería
    else if (adc_bateria<10)
        {X=120;}
    else if (adc_bateria<11)
        {X=80;}
    else if (adc_bateria>=11)
        {X=40;}

    set_adc_channel(3);               // Lectura del canal 5 -> AN0 LDR
    delay_ms(1);
    valor=read_adc();
    adc_luz=valor*388;
    valor=0;

    set_adc_channel(0);               // Lectura del canal 4 -> AN1 temperatura
    delay_ms(1);
    valor=read_adc();
    adc_temperatura=(valor*0.256)+253;
    valor=0;
    delay_ms(1);

    set_adc_channel(2);               // Lectura del canal 1 -> AN4 presión
    delay_ms(1);
    valor=read_adc();
    adc_presion= valor*7.843;
    valor=0;
    delay_ms(1);

    set_adc_channel(1);               // Lectura del canal 0 -> AN5 humedad
    delay_ms(1);
    valor=read_adc();
    adc_humedad=((valor*0.64)*0.02)+1.03;
    valor=0;
    delay_ms(1);

    output_low (BAT);                 // desconexión sensores
    output_low (SENSORES);            // desconexión test batería
}

```

```
// Enviar SMS ////////////////////////////////////////

void envio_sms(void) {

    switch(sms_devuelto) {
        case 0:
            printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Comando mal introducido%c\r\n", cod);
            printf("\r\n");
            break;

        case 1:
            printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Usuario borrado%c\r\n", cod);
            printf("\r\n");
            break;

        case 2:
            printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Usuario aceptado%c\r\n", cod);
            printf("\r\n");
            break;

        case 3:
            printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Memoria llena%c\r\n", cod);
            printf("\r\n");
            break;
    }
}
```

```

case 4:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Usuario no registrado%c\r\n", cod);
            printf("\r\n");
            break;

case 5:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            adc_temperatura=adc_temperatura-273;
            printf("\r\La temperatura es
            :%d grados%c\r\n",adc_temperatura, cod);
            printf("\r\n");
            break;

case 6:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\La presion es:
            %f%c\r\n", adc_presion, cod);
            printf("\r\n");
            break;

case 7:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\La humedad es:
            %d porcientoc%c\r\n", adc_humedad, cod);
            printf("\r\n");
            break;

```

```
case 12:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Alarma luz%c\r\n",cod);
            printf("\r\n");
            break;

case 13:    printf("\r\AT+CMGF=1\r\n");
            delay_ms(10000);
            printf("\r\AT+CMGS="+);
                for(j=0;j<11;j++)
                    {printf("%c",num[j]);}
            printf("\r\n");
            delay_ms(5000);
            cod= 0x1A;
            printf("\r\Alarma presion %c\r\n",cod);
            printf("\r\n");
            break;
        }
    }
```

## 12 Configuración y uso del Hyper Terminal

Durante la realización del proyecto se ha utilizado el HyperTerminal de Windows para supervisar la comunicación entre el módem GSM y el microcontrolador.

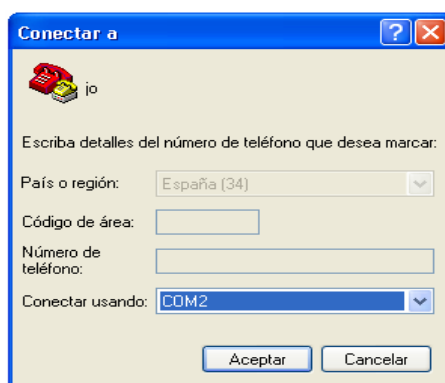
Este anexo explica como crear la sesión para realizar dicho proceso, así como la configuración de velocidades, bits de datos, bits de paridad etc...

En primer lugar, se conecta el módem a su tensión de alimentación. Una vez conectado el puerto serie entre ambos, nos dirigiremos al menú del PC y abriremos el programa HyperTerminal. Este se encuentra en *Inicio>Programas>Accesorios>Comunicación>HyperTerminal*. Seguidamente daremos nombre a la nueva conexión. Véase la **Fig. 12.1**

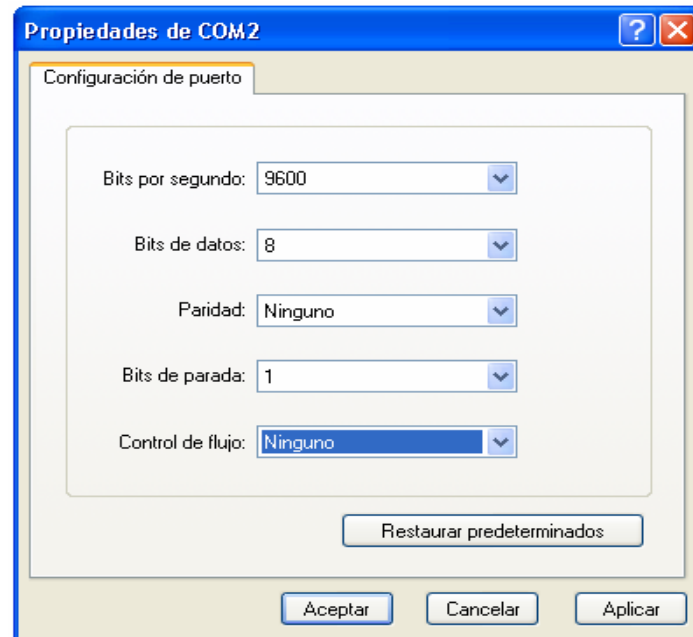


**Fig. 12.1** Inicio de comunicación con el hyperterminal

Luego seleccionaremos el puerto que estamos utilizando (COM1 o COM2) véase la **Fig. 12.2**, y a continuación visualizaremos otra ventana donde se seleccionará la velocidad de transmisión y recepción de datos, que deberá ser igual a la del módem. Véase la **Fig. 12.3**



**Fig 12.2** Selección del puerto



**Fig. 12.3** Configuración del puerto de comunicaciones

Seguidamente se pulsará en la pestaña de *propiedades>configuración para acceder a Configuración ASCII* y seleccionamos Eco de los caracteres escritos localmente. Esto nos permitirá mandar datos por el puerto serie desde el PC. Una vez hayamos realizado estos pasos ya estaremos preparados para poder interactuar con nuestro módem.

Primeros pasos a realizar: Puesta en marcha

Es muy importante que desactivemos el código PIN de la tarjeta antes de introducirla en el módem GSM. En caso contrario, la tarjeta estará esperando el código PIN y no nos permitirá poder realizar ninguna aplicación.

Abriremos la sesión del HyperTerminal en Windows como se ha explicado anteriormente.

Conectaremos la alimentación al módem. El led de estado del módem se encenderá. Posteriormente parpadeará (se está registrando en red). Una vez registrado el parpadeo será a una frecuencia diferente.

Volvemos a hyperterminal: Esperaremos unos segundos.

Teclearemos: **ATI** y pulsaremos la tecla **ENTER**.

En pantalla deberá de HyperTerminal nos aparece el siguiente mensaje referente a las características del módem.

**ATI**  
**Audiotel**  
**Base**  
**REVISION 05.00**

**OK**

Una vez realizado la comprobación de la puesta en marcha ya estamos preparados para empezar a utilizar el módem GSM.